Büroarbeit im Intranet

Diplomarbeit

TU Chemnitz
Fakultät für Informatik

eingereicht von Rico Friedrich
am Lehrstuhl für Rechnernetze und verteilte Systeme

Betreuender Hochschullehrer: Prof. Dr.-Ing. habil. Uwe Hübner
Betreuer im URZ: Dr. Wolfgang Riedel
Chemnitz, den 30. November 2000

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einführung

Im Universitätsrechenzentrum der Technischen Universität Chemnitz bestehen Geschäftsvorfälle der Büroarbeit, wie in vergleichbaren Organisationen und Unternehmen auch, größtenteils aus dem Erstellen von Geschäftsdokumenten zur Übermittlung von Informationen. Die rechnergestützte Dokumentenerstellung wird im URZ derzeit mit zwei Textsatzsystemen durchgeführt, deren Handhabung verschiedene Nachteile aufweisen, durch die sich der gesamte Geschäftsprozess behäbig gestaltet.

Durch den Einsatz neuer Technologien im Bereich Internet/Intranet ergeben sich Möglichkeiten der rationelleren Abwicklung von Geschäftsvorfällen. Dabei liegen die Schwerpunkte auf der Datenerfassung durch Web-Formulare und der Digitalisierung der Dokumente in geeigneten Strukturen zur flexiblen Weiterverarbeitung. Die Erzeugung von Geschäftsdokumenten und deren Weiterverarbeitung kann erheblich effizienter gestaltet werden. Eine Aufgabe der Digitalisierung der Verwaltung ist die Ersetzung der Papierformulare und der zugehörigen manuellen Prozesse durch elektronische Formulare und automatisierte Prozesse. Durch die Vermeidung mehrfacher Datenerfassung, Nutzung von Ausfüllhilfen und einer Validierung der Daten wird die Produktivität gesteigert und Kosten reduziert. Ein modernes elektronisches Büro kann Papier an der Quelle eliminieren, die Effizienz der Geschäftsprozesse und die Benutzerfreundlichkeit verbessern. Systemgenerierte Geschäftsdokumente können auf Vollständigkeit geprüft werden, allen relevanten rechtlichen Regelungen genügen und eine Corporate Identity wahren.

Eine beeindruckende Fallstudie liefert JetForm. Beim australischen Verteidigungsministerium richtete die Firma verschiedene elektronische Formulare ein, die nach Angaben von JetForm jährlich Kosten von 48 Millionen australischen Dollar einsparen. [Jet00] Durch die JetForm e-Formulare wurden erheblich Verarbeitungszeit eingespart, Genauigkeit und Lesbarkeit der Formulare erhöht und die mit Druck, Fehlern, Lagerung und Versand verbundenen Kosten reduziert.

Die Arbeit entwickelt und implementiert ein System zur Rationalisierung der Büroarbeit im Intranet des Universitätsrechenzentrums der TU Chemnitz. Dieses auf der Basis von Web-Formularen und XML zu erstellende System soll der elektronischen Verarbeitung von Geschäftsdokumenten dienen und manuelle Arbeitsschritte in Web-basierten elektronischen Prozessen realisieren.

Um die laufenden Geschäftsprozesse des URZ durch ein System rationeller erledigen zu können, ist eine Analyse dieser notwendig, die in Kapitel 2 durchgeführt wird. Dabei werden sieben Dokumententypen näher betrachtet, zu deren Erstellung das System vorerst dienen soll. Im dritten Kapitel wird eine Basistechnologie vorgestellt, die als Dokumentenformat und zur internen Verarbeitung zum Einsatz kommt. Es handelt sich um XML, einer Metasprache zur Definition von semantischen Strukturen. Anschließend werden vorhandene Strukturen verglichen, um eine Verwendung dieser als Dokumentenformat abzuwägen. Nach dem Entwurf des Prototypen eines modernen Web-basierten Dokumentenerstellungsystems folgt dessen Implementation, welche das sechste Kapitel erläutert. Das angefertigte System wird in Kapitel 7 dokumentiert, wobei administrative und nutzerbezogene Sichtweise getrennt werden. Das 8. und letzte Kapitel fasst die Arbeit zusammen und deutet weitere Ansätze für fortführende Arbeiten an.

Kapitel 2

Analyse laufender Geschäftsvorfälle

Die Analyse laufender Geschäftsvorfälle schafft die Grundlage für die Umstrukturierung der Geschäftsvorfälle und Neuausrichtung der Bürodokumentenerstellung im Universitätsrechenzentrum von der aktuellen Situation hin zu einem Web-basierten elektronischen Büro. Dieses soll hauptsächlich zur Erstellung und zur Bearbeitung von Bürodokumenten dienen. Der gesamte Geschäftsvorfall wird betrachtet, um einen möglichst großen Teil des Gesamtprozesses vom zu erstellenden System übernehmen zu lassen und damit einen Großteil des Geschäftsvorfalles zu optimieren. Ein Geschäftsvorfall schließt alle Ereignisse und Tätigkeiten rund um die Erstellung von Geschäftsdokumenten ein. In der Analyse wird eine eingeschränkte Anzahl häufiger Geschäftsvorfälle in Bezug auf die aktuelle Situation charakterisiert und bewertet. Dies geschieht mit dem Ziel, nachfolgend die Implementierung einer neuen Technik zur Dokumentenerstellung auf Grund der gewonnenen Ergebnisse durchführen zu können. Für die Einbindung weiterer Geschäftsvorfälle soll ebenso eine Grundlage geschaffen werden. Die Analyse wird den derzeitigen Stand der Abwicklung von Geschäftsvorfällen zeigen und auf Rationalisierungsmöglichkeiten eingehen. Dabei werden Dokumentenklassen gebildet, die sich durch charakteristische Merkmale wie Struktur und Layout auszeichnen. Eine Dokumentenklasse ist eine Gruppe gleichartiger Dokumententypen. Die zu erfassenden Dokumententypen werden im zweiten Teil dieses Kapitels näher betrachtet und später zu Dokumentenklassen zusammengefasst. Das Bilden von Dokumentenklassen hat den Zweck, den Implementationsaufwand durch Zusammenfassen ähnlicher Dokumenttypen zu verringern. Bevor die Dokumententypen einzeln betrachtet werden, wird allgemein auf Geschäftsvorfälle eingegangen.

2.1 Vorbetrachtungen zu den Geschäftsvorfällen

Viele Geschäftsvorfälle verlaufen nach gleichen Mustern, was grundlegende Gemeinsamkeiten der Dokumententypen erklärt. Geschäftsabläufe gleichen sich ebenso wie die Art der Erstellung von Dokumenten. Unterschiedliche Dokumententypen, z.B. verschiedene Arten von Briefen, nutzen das selbe Layout. Da diese und weitere Übereinstimmungen und Konzepte der Geschäftsprozesse öfter auftreten, wird nachfolgend auf sie eingegangen. In der später folgenden detaillierten Analyse einzelner Dokumenttypen können sie dann referenziert werden. Grundlegende Gegebenheiten, wie das Einhalten von bestimmten Normen, gilt es einzuhalten oder zu nutzen. Des Weiteren werden die systemtechnischen und technologischen Aspekte des realen Einsatzes im Universitätsrechenzentrum beleuchtet. Die Vorbetrachtung spricht in den nächsten Abschnitten allgemein zutreffende Aspekte der Geschäftsvorfälle an.

2.1.1 Mitarbeiter

Da Geschäftsvorfälle im Rechenzentrum nur von Mitarbeitern des URZ angestoßen werden, haben auch nur sie die Aufgabe und das Recht Geschäftsdokumente des URZ anzufertigen. Nach einer kurzen Klärung des Begriffes eines Mitarbeiters im URZ wird näher auf das Mitarbeiterkonzept zur Erstellung von Dokumenten eingegangen.

Mitarbeiter des URZ sind alle Personen, die das Universitätsrechenzentrum beschäftigt. Sie sind befristet oder unbefristet an der Universität angestellt und können dem Telefon- und Mailverzeichnis[1] des Rechenzentrums entnommen werden. Eine Liste der aktuell angestellten Mitarbeiter ist im Anhang A zu finden. Darin ist auch das Kürzel jedes Mitarbeiters enthalten, das in Dokumenten wenn nötig im Kontaktinformationsbereich angegeben wird. Es ist meist mit dem Nutzerkennzeichen identisch oder bei langen Nutzerkennzeichen eine verkürzte Form dessen.

Derzeitig sind alle analysierten Dokumente von jedem Mitarbeiter im dafür vorgesehenen Verzeichnis les- und schreibbar. Im allgemein häufigsten Fall wird im Sekretariat ein Dokument für einen Mitarbeiter erstellt, der den Inhalt des Dokumentes liefert und meist auch unterschreibt. Es gibt aber auch den Fall, dass Mitarbeiter außerhalb des Sekretariates ihre Dokumente eigenhändig schreiben. Im Sollzustand ist nicht vorgesehen und auch nicht nötig, dass jeder Mitarbeiter beliebige Dokumente erstellen und einsehen kann. Auf Grund dieser Tatsache sollte es ein zukünftiges Einschränken der Rechte zur Dokumentenerstellung geben. Da es Mitarbeiter gibt, die sich Dokumente vom Sekretariat erstellen lassen, könnten zwei Gruppen aus dieser Konstellation abgeleitet werden. Im Brief wird dieses Verhältnis durch das Erscheinen beider Kürzel dargelegt, wenn Personen beider Gruppen am Erstellen beteiligt sind. Allerdings lohnt sich eine Erstellung von Berechtigungsgruppen nicht, weil beide Gruppen zum Editieren des Dokumentes Bearbeitungsrechte brauchen. Ein weiterer Ansatz ist das Bilden einer Berechtigungsgruppe für die Mitarbeiter, die das Recht zum Unterschreiben von Dokumenten innehaben. Bei näherer Betrachtung verflüchtigt sich diese Idee, da die Unterschrift bei digitalen Dokumenten auf einer digitalen Signatur basieren muss, aber quasi alle Dokumente in Papierform das Sekretariat verlassen. Das dennoch interessante Thema der digitalen Signatur wird später tiefgründiger betrachtet.

Abschließend kann festgehalten werden, dass das zu erstellende System über eine Nutzerverwaltung verfügen sollte, die für jeden Mitarbeiter den Zugriff auf einzelne Dokumententypen definiert.

2.1.2 Typische Geschäftsabläufe

In der Verwaltung des Universitätsrechenzentrums können zwei typische Geschäftsabläufe zum Erstellen von Dokumenten beobachtet werden. Ein Geschäftsablauf im Sinne der Diplomarbeit ist die Menge aller nötigen Schritte vom Wunsch, ein Geschäftsdokument zu senden, bis zur Fertigstellung der Mitteilung. Die Geschäftsabläufe werden folgend sowohl personen- als auch systemunabhängig dargestellt, da dies gesondert zum jeweiligen Dokumententyp betrachtet wird.

Variante I

Ein Mitarbeiter außerhalb des Sekretariat möchte ein Schreiben an einen Empfänger senden. Dazu kontaktiert er das Sekretariat per Email oder mündlich und übermittelt seine Wünsche. Das Personal im Sekretariat verfasst das Schreiben und druckt es aus. Die Unterschrift wird beim Mitarbeiter selbst oder per Unterschriftenmappe eingeholt, der das Dokument auf Korrektheit überprüft. Es besteht auch die Möglichkeit, dass die Sekretärin im Auftrag unterschreibt. Die entstandene Post wird von einer Sekretärin zur Poststelle der TU Chemnitz gebracht. Dieser Fall stellt die Regel dar.

[1] Telefon- und Mailverzeichnis des URZ: http://www.tu-chemnitz.de/urz/leitung/tel-urz.html

Variante II

Einige wenige URZ-Mitarbeiter schreiben ihre Dokumente selbst. Bis auf eine Ausnahme werden die Dokumente jedoch wieder im Sekretariat gedruckt, weil dort das Papier mit dem Layout des URZ lagert. Das nötige Einholen der Unterschrift kann wiederum wie oben geschildert geschehen. Eventuell könnte der Mitarbeiter auch eine eingescannte Unterschrift einfügen, die ein nochmaliges Unterschreiben und damit den entsprechenden Weg überflüssig macht. Die entstandene Post wird von einer Sekretärin zur Poststelle der Universität getragen.

Schlussfolgerung

In Bezug auf die benötigte Zeit zum Erstellen eines Dokumente ist Variante II die wesentlich Effizientere von beiden. Es vergeht weniger Zeit vom Wunsch bis zum Fertigstellen eines Dokumentes, auch auf Grund nicht auftretender Kommunikationsschwierigkeiten. Das liegt letztlich daran, dass zum Erstellen eines Dokumentes weniger Arbeitsschritte nötig sind. Letztendlich wird die entstandene Post zweimal am Tag der Poststelle der TU Chemnitz übermittelt. Die benötigte Zeit vom Entstehen des Wunsches einer Briefversendung bis zur Ablieferung des Briefes in der Poststelle beträgt damit im Durchschnitt 1/2 Tag. Dies ist für den Zweck ausreichend, stellt sich aber als „Flaschenhals“ beim Versuch einer zeitlichen Optimierung heraus. Eine Web-basierte Lösung kann die Gesamtzeit eines vollständigen Geschäftsvorfalles nicht reduzieren, wohl aber die entscheidende Zeit für das Erstellen eines einzelnen Dokumentes. Der Gang zum Mitarbeiter, um dessen Unterschrift einzuholen, kann auch eingespart werden.

Die Möglichkeit, eine eingescannte Unterschrift zu benutzen, anstatt per Hand zu unterschreiben, bildet zur Zeit eine Sicherheitslücke. Die gespeicherten Dokumente sind für alle Mitarbeiter lesbar und sie könnten somit auf die vorhandenen eingescannten Unterschriften zugreifen. Bei Kompromittierung eines Accounts kann auf alle eingescannten Unterschriften zugegriffen werden. Auf Grund dieser Mängel wird eine Verwendung der eingescannten Unterschrift nicht weiter unterstützt.

Variante II erscheint insgesamt effizienter als Variante I. Das liegt daran, dass bei der ersten Variante die beteiligten Personen abwechselnd involviert sind und somit der Aufwand für Kommunikation und Kontaktherstellung steigt. Manchmal ist es auch nötig, dass mehrere Personen das Schreiben einsehen müssen, um diese inhaltlich und fachlich abzustimmen. Eine Online-Beratung mit sofortiger Möglichkeit zur Änderung der Dokumente erscheint hier als effektiver Mehrwert eines Web-basierten Systems.

Um die Geschäftsabläufe zu rationalisieren, sollen einzelne Arbeitsschritte eingespart und durch neue Techniken der Erstellung ersetzt werden. Dies muß aber auch unter dem Aspekt der gesetzlichen und technischen Möglichkeiten geschehen, die ab dem Abschnitt 2.1.3 näher erläutert werden.

2.1.3 Technische Gegebenheiten und Software zur Dokumentenerstellung

Nach einer Beschreibung der technischen Ausstattung des URZ wird auf die verwendete Software zum Erstellen von Dokumenten eingegangen.

Die Dokumente werden in der Verwaltung des Universitätsrechenzentrums an PC's erstellt. Alle Rechner sind an das lokale Universitätsnetzwerk angeschlossen und können auf die AFS-Zelle der Universität zugreifen. Drucker befinden sich im Sekretariat und vereinzelt in Mitarbeiterbüros. Auf den Rechnern ist wahlweise oder parallel das Betriebssystem RedHat Linux 6.0 und/oder Windows NT 4.0 installiert. Zur Erstellung der Dokumente werden die Texterstellungssoftware LaTeX und StarOffice verwendet, auf die sich die folgenden Anschnitte näher beziehen.

Dokumentenerstellung mit StarOffice

StarOffice ist eine komplexe Software zum Erstellen von Bürodokumenten. Es bietet Fähigkeiten zur Textverarbeitung, zur Tabellenkalkulation und zur Gestaltung von Grafiken und Präsentationen und enthält eine eigene Datenbank. Im Rechenzentrum wird StarOffice in der Version 5.2 verwendet.

Beim Erstellen von Geschäftsdokumenten werden StarOffice-Vorlagen als Musterdokumente verwendet. Die Vorlagen wurden mit StarOffice erstellt und können verschiedenste Funktionalitäten, wie zum Beispiel die eines Serienbriefes enthalten. Auf sie kann jeder Mitarbeiter lesend unter `/afs/tu-chemnitz.de/urz/buero/muster/StarOffice/` zugreifen. StarOffice-Dokumente werden in einem eigenen Binärformat gespeichert. Das Hauptverzeichnis zum Ablegen der fertigen StarOffice-Dokumente lautet `/afs/tu-chemnitz.de/urz/buero/`.

Die Benutzung eines StarOffice-eigenen Datenbanksystems ist auch möglich. Bei der Verwendung von StarOffice unter NT und Linux wird auf eine Firmen-Adress-Datenbank zugegriffen.

Dokumente werden mit StarOffice sowohl unter NT als auch unter RedHat Linux erstellt. Gelegentlich gibt es Schwierigkeiten bei der Darstellung von speziellen Zeichen bei unter NT erstellten Dokumenten, die unter Linux gelesen oder weiter bearbeitet werden. Um dies zu umgehen, bietet sich eine Web-basierte Lösung an, da diese unabhängig vom Betriebssystem funktioniert. Mit StarOffice werden im Rechenzentrum Dokumententypen wie Angebotseinholungen, Faxe, Zugangsberechtigungen, Beschaffungsanträge, Briefe und Gutachten angefertigt.

Verwendung der Adressdatenbank

Wie schon erwähnt, wird bei der Erstellung unter StarOffice auf eine eigene Adressdatenbank zugegriffen. Sie beinhaltet die Adressen von Firmen, die aus Elementen wie Firma, Abteilung, Name, Telefon und weiteren bestehen. Die Einträge sind je nach Notwendigkeit teilweise oder vollständig gefüllt. Eine Firma kann mehrere Datenbankeinträge besitzen, wenn zum Beispiel verschiedene Ansprechpartner in dieser Firma angeschrieben werden sollen. Die StarOffice-Adressdatenbank wurde in eine MySQL-Datenbank übertragen, sodass vom zu erstellenden System benötigte Adressdaten aus dieser gewonnen werden können.

Dokumentenerstellung mit LaTeX

Eine weitere Software, die im Universitätsrechenzentrum Einsatz findet, ist LaTeX, ein mächtiges Textsatzsystem [Rie95]. Es eignet sich für die Erstellung von wissenschaftlichen Dokumentationen, einfachen Briefen bis hin zu kompletten Büchern. LaTeX ist ein Makropaket von Leslie Lamport und setzt auf TeX auf, einem Programm von Donald E. Knuth, welches zum Setzen und Drucken von Texten und mathematischen Formeln dient.

Für die Dokumententypen Brief, Hausmitteilung und Gutachten gibt es eine Mustervorlage. Diese Vorlagen liegen im Verzeichnis `/afs/tu-chemnitz.de/urz/buero/muster/latex` und sind für alle URZ-Mitarbeiter lesbar. Sie werden zur Erstellung neuer Dokumente verwendet. Hierbei wird nicht nur der Inhalt des Schreibens eingetragen, sondern es müssen auch die notwendigen LaTeX-Befehle beherrscht werden. Zur Vereinfachung wurden für manche Dokumente neue LaTeX-Dokumentenklassen definiert, deren spezieller Befehlsatz dokumentiert und überschaubar ist. Die Klassen befinden sich physisch unter `/usr/share/texmf-local` auf jedem Rechner der zum URZ gehört und unter Linux betrieben wird. Somit hat jede Person Zugriff auf diese Klassen, die sich an einem Rechner im URZ einloggen kann. Das Speichern der Geschäftsdokumente erfolgt im selben Verzeichnis wie bei StarOffice. Unter LaTeX ist keine Datenbank verfügbar, aus der Adressdaten abgegriffen werden könnten. Alle Daten müssen gesucht und per Hand eingetragen werden.

Zusammenfassung

Außer dem schreibgeschützten Vorlagenverzeichnis ist es jedem Mitarbeiter möglich, erstellte Dokumente in Verzeichnissen zu sichern, für die er eine Schreibberechtigung hat. Das beliebige

Speichern ist damit ein Nachteil für beide Systeme, denn Mitarbeiter können ihre Dokumente selbst benennen und eventuell ungewollt in falschen Verzeichnissen sichern. Ein modernes System sollte hier Verzeichnisse zum Speichern und Namenskonventionen vorgeben.

Die AFS-Rechte für die Verzeichnisse in denen die Dokumente lagern wurden so gesetzt, dass jeder Mitarbeiter auf alle Dokumente im Dokumentenverzeichnis zugreifen kann. Obwohl sich daraus ergibt, dass die analysierten Dokumenttypen nicht unbedingt vor anderen Mitarbeitern geheim gehalten werden müssen, ist es dennoch ein weiterer Nachteil, der durch Berechtigungsgruppen gelöst werden kann. Geschäftsdokumente, die nur von einen speziellen Kreis von URZ-Mitarbeitern erstellt werden sollen, könnten damit vor unberechtigtem Zugriff geschützt werden.

Da alle Dokumente eines Jahres in einem Verzeichnis aufgenommen werden, ist das Wiederfinden von Dokumenten erschwert. Laut existierender Namenskonvention enthalten Dateinamen das Nutzerkennzeichen des Erstellers, was das Suchen der Dokumente im Verzeichnis erleichtert. Allerdings ist das Suchen nach Dokumenteninhalten mit dem Öffnen aller namentlich in Frage kommender Dateien verbunden. Mit der Bereitstellung einer entsprechenden Funktionalität kann ein neues System auch hier Abhilfe schaffen.

StarOffice kann auf eine eigene Adressdatenbank zugreifen, was ein entscheidender Vorteil gegenüber der Dokumentenerstellung mit LaTeX ist. In Verbindung mit einer Datenbank bietet StarOffice eine einfache Erstellung von Serienbriefen. In LaTeX ist dafür ein vielfach größerer Zeitaufwand erforderlich, da alle Adressen gesucht und von Hand eingetragen werden müssen.

In der Bedienbarkeit überzeugt StarOffice mit einer grafischen Oberfläche, dem Erstellen von Vorlagendokumenten, einer Datenbankintegration und weiteren Funktionalitäten wie die Erstellung von Serienbriefen. Dies alles vereinfacht die Erstellung von Dokumenten. LaTeX-Dokumente hingegen werden unter Verwendung von LaTeX-Befehlen geschrieben. Es gibt für einige Dokumente spezielle LaTeX-Dokumentenklassen, die durch einen kleinen Satz optimierter Befehle die Dokumentengenerierung vereinfachen. Trotzdem ist hier, wenn auch nur oberflächlich, wiederum LaTeX-Programmierkenntnis erforderlich. StarOffice ist hier eindeutig effizienter als das mächtige, aber eher schwierig bedienbare LaTeX.

Die Textgestaltung, die mit beiden Werkzeugen stark beeinflusst werden kann, steht allgemein für die Formatierung und die Anordnung des Textes einer Mitteilung. Für die Geschäftsdokumente des URZ sind sie sogar zu reichhaltig, da eine freie Gestaltung die Corporate Identity gefährden kann. Sie lassen dem Erstellenden zu viel Spielraum und sollten in einem entsprechend entwickelten Werkzeug eingeschränkt werden.

Obwohl die Rechner im Rechenzentrum relativ neu sind, ist StarOffice der enorme Verbrauch von Rechnerresourcen anzumerken und fällt negativ auf. Die Verfügbarkeit aller LaTeX-Dokumentenklassen auf allen Rechnern des URZ legt unnötig URZ-interne Layouts zur Dokumentenerstellung offen und könnte durch eine zentrale Lösung eliminiert werden. StarOffice wurde später eingeführt und ist auf Grund der angeführten Vorteile gegenüber LaTeX inzwischen öfter in Benutzung. Das Ziel der Arbeit ist die Erstellung eines Tools, das die angeführten Probleme beseitigt und sich an vorhandener Funktionalität, wie der Erstellung von Serienbriefen, orientiert.

2.1.4 Normen für den Geschäftsverkehr

Um die genormte Verfassung und damit die Korrektheit der erstellten Dokumente sicherzustellen, werden die deutschen Normen DIN 676 [DIN676] und DIN 5008 [DIN5008] herangezogen.

Die Norm DIN 676 legt einheitliche Einzelvordrucke für Geschäftsbriefe fest, um den Schriftverkehr zu erleichtern. Die Festlegungen der Norm gelten für das Beschriften durch Textverarbeitungssysteme und Schreibmaschinen. In dieser Norm werden zwei Formen für Geschäftsbriefe unterschieden und Hinweise zu Bezeichnungen, Anordnung der Felder, Kommunikations- und Geschäftsangaben präsentiert. Im URZ wird es meist um den „Geschäftsbrief DIN 676 – B-A4“ gehen. Dies ist die Bezeichnung für einen Einzelvordruck der Form B für einen Geschäftsbrief im Format A4.

Die Bestimmungen der Norm DIN 5008 tragen dazu bei, die Texteingabe zu erleichtern, Schreibarbeit einzusparen, eine Verarbeitung der Informationen zu ermöglichen und die Übertragung der Daten zwischen unterschiedlichen Geräten sicherzustellen. Die Norm legt fest, wie durch ein einheitliches Anwenden von Schriftzeichen bei Schreibmaschinen und Textverarbeitungssystemen mit alphanummerischen Tastaturen eine leichte und eindeutige Lesbarkeit der Schrift gesichert werden und wie durch entsprechende Gestaltungsvorschriften die Schriftstücke zweckmäßig und übersichtlich gestaltet werden können.

Einige grundlegende und entscheidende Regeln für die Dokumentenerstellung werden nachfolgend anlehnend an DIN 5008 angesprochen. Alle erwähnten Richtlinien sind im Abschnitt 12 der Norm definiert.

1. Die Variation der Schriftgröße ist zulässig. Schriftgrößen unter 10 Punkten und ausgefallene Schriftarten sind zu vermeiden.

2. In der Regel werden Briefe in einer einheitlichen Schriftgröße geschrieben.

3. Ab der zweiten Seite ist ein Aufdruck nicht mehr nötig.

4. Eine fortlaufende obige Nummerierung wird von der zweiten Seite eines Briefes an gefordert. Genaue Positionen und weitere Vorschriften zur Seitennummerierung auf Blättern ohne Aufdruck kann dem Abschnitt 12.14 entnommen werden.

Falls Änderungen der Erstellungsart in Bezug auf Struktur der Dokumente angebracht sind, dann wird das zur jeweiligen Dokumententypanalyse diskutiert.

2.1.5 Bilden von Layouttypen

Es gibt zwei mehrfach genutzte Layouts in den betrachteten Dokumenten, die in den folgenden Abschnitten aufgezeigt werden. Um sie zu unterscheiden, werden sie in das „Layout für ausgehende Post“ und das „Layout für interne Post“, dem internen Geschäftsverkehr, klassifiziert. Die Layouttypen werden gebildet, um neben der Struktur der Dokumente einen weiteren Anhaltspunkt zur Klassifizierung der Dokumententypen zu erhalten. Der Brief, die Angebotseinholung und die Zugangsberechtigung nutzen das Layout für ausgehende Post. Das Fax hingegen enthält nur den Kopf des Layouts für ausgehende Post. Die Hausmitteilung und das Gutachten nutzen das Layout für internen Geschäftsverkehr. Außerdem hat der Beschaffungsantrag ein eigenes Layout, welches bei der Analyse des Beschaffungsantrages selbst erläutert wird. Wenn die LaTeX-Dokumentenklassen das Layout automatisch generieren, werden die Layoutvorlagen nicht mehr benötigt. (ähnlich der Erstellung von Dokumenten mit StarOffice)

Ausgehende Post

Am 24.09.1997 wurde vom Kanzler der TU Chemnitz im Rundschreiben 31/97 eine einheitliche Neugestaltung der Briefköpfe festgelegt. Demnach besteht ein Dokument für ausgehende Post aus Kopf, Körper und Fuß. Dieses Layout richtet sich nach DIN 676 für Geschäftsbriefe und den Schreib- und Gestaltungsregeln für die Textverarbeitung entsprechend DIN 5008. Der Kopf ist in Abbildung 2.1 verkleinert zu sehen; der Fuß in Abbildung 2.2.

Abbildung 2.1: Briefkopf für ausgehende Post

Dienstsitz Sekretariat URZ:
Straße der Nationen 62
Raum 357c
Telefon: (0371) 531-1551
Fax: (0371) 531-1629
Email: urz@tu-chemnitz.de

Postanschrift:
TU Chemnitz
Universitätsrechenzentrum
D-09107 Chemnitz

Paketanschrift:
TU Chemnitz
Universitätsrechenzentrum
Straße der Nationen 62
D-09111 Chemnitz

Bankverbindung der Universität:
Empfänger: Landesoberkasse Chemnitz
Sparkasse Chemnitz
BLZ: 870 500 00
Konto-Nr.: 355 000 1800

Abbildung 2.2: Brieffuß für ausgehende Post

Interne Post

Das Layout der Dokumente für interne Post besteht nur aus Kopf und Körper und fällt nicht unter die Rubrik Geschäftsbriefe. Allerdings sind die Schreib- und Gestaltungsregeln für die Textverarbeitung nach DIN 5008 zu beachten. Für den internen Briefverkehr gibt es keine Layoutanweisung. Der Kopf ist in verkleinerter Form in Abbildung 2.3 dargestellt.

TU Chemnitz, URZ

Hausmitteilung

Abbildung 2.3: Briefkopf für interne Post

2.1.6 Entwicklung der Erzeugung von Bürodokumenten

Seit dem Einsatz von Computern wurden Geschäftsdokumente mit LaTeX unter UNIX auf Sun Workstations erstellt. Mit dem Umrüsten auf Intel-Rechner Anfang 1999 betrieb das Rechenzentrum die Betriebssysteme Windows NT 4.0 und RedHat Linux parallel. Unter Linux erstellte man Dokumente weiterhin mit LaTeX. Zusätzlich wurde StarOffice unter beiden Betriebssystemen eingeführt, da sich damit zum Beispiel Serienbriefe einfacher erstellen lassen.

Zur Erleichterung der Erstellung von Bürodokumenten mit LaTeX wurden von Herrn Dr. W.Riedel LaTeX-Dokumentenklassen zur Verfügung gestellt. Diese Klassen sind auf jedem Rechner im URZ lokal unter `/usr/share/texmf-local/tex/latex/urz/` abgelegt. Zum Erzeugen von Dokumenten können sich die Mitarbeiter an Mustern orientieren. Mit der Einführung von StarOffice wurden Vorlagen geschaffen, die bei jeder Erstellung der analysierten Dokumententypen verwendet werden. Des Weiteren erleichtert die angelegte StarOffice-interne Adressdatenbank die Eingabe von Adressinformationen.

In letzter Zeit wurden Versuche unternommen, das Web bei der Erstellung von Bürodokumenten einzusetzen. Im folgenden Abschnitt wird ein Werkzeug beschrieben, welches einen Ansatzpunkt für den Implementierungsteil der Diplomarbeit sein könnte. Es ist ein Werkzeug zur Erstellung von Formularen über ein Web-Interface, welches auch Ausgangspunkt für weiterführende Arbeiten von Herrn Dr. W.Riedel war. Es entstanden Werkzeuge zur Erstellung von Bürodokumenten, die aber nie zum Einsatz kamen, aber trotzdem als eine Quelle für weitere Implementierungsideen gesehen werden.

Das Werkzeug zur Generierung von Beschaffungsanforderungen

Es existiert bereits ein elektronisches Web-basiertes und funktionstüchtiges Werkzeug, mit welchem Dokumente unter Nutzung eines Web-Interfaces erstellt werden können. Hierbei handelt es sich um das Werkzeug zur Generierung von Beschaffungsanforderungen an den Kanzler der TU Chemnitz [And98] von Herrn Dr. J.Anders. Dieses Werkzeug stellt eine Schnittstelle in Form eines HTML-Formulars bereit, welches alle benötigten Eingaben gleich dem tatsächlichen großen Kanzlerbeschaffungsformular aufnehmen kann. Das eigentliche Dokument wird dann mit dem

Werkzeug im PostScript-Format erstellt und kann auf den PostScript-Druckern der Universität ausgedruckt werden. Als Voraussetzung muß ein Tool zur Anzeige von PostScript-Dokumenten installiert sein. Außerdem sollte der benutzte Web-Browser beim Eintreffen von PostScript-Daten den Aufruf des PostScript-darstellenden Tools veranlassen. Eine Verbesserung bietet die aktuelle Version 2, die es erlaubt, ein ausgefülltes Formular als HTML-Datei abzuspeichern. Zu einem späteren Zeitpunkt kann die gespeicherte Datei in einen Web-Browser geladen werden, um Änderungen oder Ergänzungen im HTML-Formular vornehmen zu können. Diese Erweiterung wurde von Herrn Dr. R.Vogel programmiert.

Das Werkzeug zur Erzeugung eines PostScript-Dokumentes basiert auf einem CGI-Programm, welches mit der Programmiersprache C geschrieben wurde. Die Formulareingaben werden durch das Anklicken des Submit-Buttons an ein CGI geschickt. Vom CGI empfangenen Daten werden in ein mit LaTeX vorgefertigtes Gerüst eingefügt und per LaTeX in ein dvi-File übersetzt, welches seinerseits in ein PostScript-Dokument konvertiert wird. Das System zur Generierung von großen Kanzlerbeschaffungsanforderungen ist praxistauglich. Ein Formular kann schneller als mit der Schreibmaschine erstellt werden, sieht den mit Schreibmaschine ausgefüllten Formularen täuschend ähnlich und übertrifft diese natürlich in ihrer sauberen Ausführung. Als ein Vorteil stellt sich die Möglichkeit zum Ändern der erstellten Formulare heraus. Weiterhin ist dieses System im Gegensatz zu anderer Office-Software unabhängig vom Betriebssystem und im gesamten Campusnetz verfügbar. Die Speicherung als HTML-Datei ist eine einfache Lösung zur späteren Änderung oder Ergänzung, die hier auch ausreichend ist. Um eine Struktur hinter den Formulardaten erkennen zu wollen, ist HTML unzureichend. Auf Grund der in der Universität in einer Vielzahl vorhandenen PostScript-Druckern ist die Erstellung eines PostScript-Dokumentes angebracht. Dieses Tool bietet gute Ansätze, die bei der Realisierung in Betracht gezogen werden sollten.

2.2 Analyse der Dokumententypen

Um eine exakte Erzeugung der Dokumententypen im neuen System zu garantieren, müssen die charakteristischen Eigenschaften der zu erstellenden Dokumente detailliert aufgenommen werden. Dazu werden Verwendungszweck, Geschäftsablauf, involvierte Personen, Erstellungsvorgang, Struktur, Inhalt und Formatierung, Layout und mögliche weitere Besonderheiten jedes Geschäftsvorfalles betrachtet. Dabei ist der Aufbau der Dokumente für eine spätere Klassifizierung von größter Bedeutung. Im Folgenden werden die Dokumententypen Brief, Angebotseinholung, Fax, Hausmitteilung, Stellungnahme zur Beschaffungsanforderung (Gutachten), Zugangsberechtigung und Beschaffungantrag analysiert. In den Abbildungen zu jedem Dokumententyp werden zukünftige Strukturen dargelegt, die auch Veränderungen zu den derzeit erstellten Dokumenten aufweisen können. In der Analyse der einzelnen Dokumententypen sind angegebene Verzeichnispfade relativ zu `/afs/tu-chemnitz.de/urz/` zu sehen.

2.2.1 Der Brief

Der Zweck eines Briefes ist es, eine Mitteilung an eine Firma zu senden. Die Mitteilung kann zum Beispiel den Eingang einer Lieferung bestätigen oder eine Reklamation enthalten. Briefe gehören zu den am meisten erstellten Dokumenten. Zur Brieferstellung kommen beide in Abschnitt 2.1.2 vorgestellten Varianten ohne Einschränkung zur Anwendung. Es ist grundsätzlich jeder URZ-Mitarbeiter berechtigt, einen Brief zu verfassen.

Zum Schreiben des Briefes werden beide im Abschnitt 2.1.3 angesprochenen Varianten genutzt. Briefe können mit StarOffice unter Zugriff auf die Adressdatenbank oder unter LaTeX erstellt werden. Folgenden Angaben nennen verwendete Erstellungshilfen und den Speicherort für Briefe.

LaTeX	Vorlage	buero/muster/latex/urzbrief1.tex
	Dokumentenklasse	urzbrief1.cls
StarOffice	Vorlage	buero/muster/StarOffice/urzbrief.vor
Beide	Speicherpfad	buero/briefe/

Es existiert ein HTML-Formular zur Erzeugung von Briefen per Webschnittstelle. Dieses ist auf Grund zu geringer Funktionalität nicht zum Einsatz gekommen und basiert auf dem in Abschnitt 2.1.6 vorgestellten Tool von Herrn Dr. J.Anders.

Struktur und Inhalt

Der Dokumententyp Brief (Abbildung 2.4) ist nach DIN 676 Vordruck Form B aufgebaut. Der Körper des Schreibens enthält die Postanschrift des Absenders. Diese ist unveränderlich und lautet

„Technische Universität Chemnitz, Universitätsrechenzentrum, 09107 Chemnitz“.

Die Anschrift des Empfängers besteht aus Firma, Abteilung, Vorname, Name, Straße und Nummer, Postleitzahl und Ort. Im Kommunikationsbereich befinden sich die Leitwörter:

- `Ihre Zeichen, Ihre Nachricht vom`
- `Unser Zeichen, unsere Nachricht vom`
- `Telefon`
- `Chemnitz, den`

Laut DIN 676 besteht das dritte Feld aus `Telefon, Name` und das Vierte heißt `Datum`. Nach einer optionalen Betreffzeile folgt die Anrede, an die sich ein im Sinne der Normen variabler Text anschließt. Im Textbereich können fette oder kursive Schriftzüge, Tabellen und Listen enthalten sein. Der letzte Strukturabschnitt besteht aus einem Gruß und zwei Unterschriftszeilen, die in der Regel den Namen und den Arbeitsbereich des Briefautors enthalten. Für den Brief wird Layout für ausgehende Post aus Abschnitt 2.1.5 verwendet.

2.2.2 Die Angebotseinholung

Eine Angebotseinholung wird als Brief an eine oder mehrere Firmen gesandt, um ein Angebot mit konkreten Preis- und Lieferterminangaben für die in der Anfrage angeführte Artikel oder Leistungen zu erhalten. Beide im Abschnitt 2.1.2 angesprochenen Geschäftsabläufe kommen bei diesem Dokumententyp zur Anwendung. Angebotseinholungen dürfen vom Sekretariat auch im Auftrag unterschrieben werden.

Das Schreiben wird sowohl unter Linux, als auch unter Windows NT mit StarOffice erstellt, meist als Serienbrief verfasst und dabei auf die StarOffice-Funktionalität zum Erzeugen von Serienbriefen zurückgegriffen. Hierbei wird eine Adressen-Datenbank genutzt, die die Adressen der anzuschreibende Firmen enthält. Es können die schon erwähnten Probleme der Darstellung von Zeichen zwischen StarOffice unter Windows NT und StarOffice unter Linux auftreten.

LaTeX	Vorlage	buero/muster/latex/urzangebot.tex
	Dokumentenklasse	urzangebot.cls
StarOffice	Vorlage	buero/muster/StarOffice/angebot.vor
Beide	Speicherpfad	buero/angebot/

Wie beim Brief gibt es hier auch ein unbenutztes HTML-Formular zur Erstellung von Angebotseinholungen per Webschnittstelle.

Abbildung 2.4: Struktur des Briefes

Struktur und Inhalt

Der Dokumententyp Angebotseinholung (Abbildung 2.5) ist nach DIN 676, Vordruck Form B, mit einer besonderen Anordnung der Kommunikationsangaben nach DIN 676 Bild 5 aufgebaut. [DIN676] Die Postanschrift des URZ und die Anschrift des Empfängers stimmen mit der des Briefes überein. Derzeit sind im Bereich für Kommunikationsangaben die Leitwörter

- `Bearbeiter`,
- `Telefon`,
- `Fax` und
- `Datum`

angeführt, werden aber noch durch einen Eintrag für die `e-Mail` ergänzt. Der fett hervorgehobenen Wortlaut des Betreffs lautet `Angebotseinholung`. Der Anrede folgt üblicherweise der Text

> *„bitte unterbreiten Sie uns bis [Datum] ein Angebot über:“.*

Dieser Text ist variabel und kann durch einen geeigneteren, der Situation entsprechenden Absatz ersetzt werden. Unter diesem, um ein Angebot bittenden Text, werden in Tabellenform die gewünschten Positionen angegeben. Ein Eintrag besteht aus

1. *Positionsnummer,*
2. *Stückzahl und*
3. *Artikelbeschreibung.*

Wahlweise kann auch noch ein Feld mit Punkten folgen, in welches der Anbieter zur vereinfachten Geschäftsabwicklung den Preis einträgt. Nach der Auflistung der Artikel können weitere Bemerkungen stehen. Vor dem Gruß befindet sich ein fester Absatz, der folgende Angaben enthält:

> *Weitere Informationen sind erwünscht:*
>
> 1. *Angaben zur Lieferzeit*
> 2. *Transport- und sonstige Kosten*
> 3. *Angaben zur Gewährleistung*
> 4. *Zahlungsbedingungen (Skonto, Hochschul- oder Mengenrabatt)*

Der Grußbereich orientiert sich an dem des Briefes. Im Textbereich ist wiederum kursiver und fettgeschriebener Text zulässig. Außerdem werden die Artikel in Tabellenform dargestellt. Die Angebotseinholung verwendet das Layout für ausgehende Post und entspricht exakt dem des Briefes.

2.2.3 Das Fax

Ein Fax-Dokument wird geschrieben, um eine Nachricht an eine Firma per Fax zu übermitteln. Es ist natürlich auch möglich, Faxe universitätsintern zu versenden. Dieser Fall steht zwar nicht im Vordergrund, wird aber auch betrachtet.

Der Ablauf des ursprünglich angedachten Geschäftsvorganges zur Erstellung von Faxen kann am besten mit Variante I aus Abschnitt 2.1.2 beschrieben werden. Allerdings gibt es, wie im nächsten Abschnitt erläutert wird, einen Fax-Dienst des URZ, der einen Geschäftsablauf ähnlich Variante II zu Folge haben kann.

Die primäre Faxerstellung, die von einer StarOffice-Vorlage unterstützt wurde, ist nur noch selten im Sekretariat im Gebrauch. Es gibt auch eine nicht verwendete Vorlage für Faxe unter LaTeX.

Feld für Briefkopf: Briefkopf für ausgehende Post

TU Chemnitz, Universitätsrechenzentrum, 09107 Chemnitz

Feld für Anschrift des Empfängers

Bearbeiter
Telefon
Telefax
e-Mail
Datum

Angebotseinholung

Anrede

Aufforderung zur Angebotsunterbreitung

Position1	Stückzahl	Beschreibung
Position2	Stückzahl	Beschreibung
Position3	Stückzahl	Beschreibung

Bemerkungen

Weitere Informationen sind erwünscht:
1. Angaben zur Lieferzeit
2. Transport- und sonstige Kosten
3. Angaben zur Gewährleistung
4. Zahlungsbedingungen (Skonto, Hochschul- oder Mengenrabat)

Gruß

Feld für Geschäftsangaben: Brieffuß für ausgehende Post

Abbildung 2.5: Struktur der Angebotseinholung

LaTeX	Vorlage	buero/muster/latex/urzfax.tex
	Dokumentenklasse	urzfax.cls
StarOffice	Vorlage	buero/muster/StarOffice/urzfax.vor
Beide	Speicherpfad	buero/briefe/

Zur Zeit ist die übliche Methode zum Erstellen und Versenden von Faxen die Verwendung des Fax-Dienstes der URZ. Dieser wurde von Herrn F.Richter und Herrn A.Mittelbach erstellt und wird jetzt von ihnen verwaltet. Das System zeichnet sich durch drei unterschiedlichen Methoden aus, um ein Fax zu versenden.

1. Durch eine Web-Schnittstelle kann ein LaTeX-Dokument spezifiziert werden. Faxe werden oft wie Briefe mit LaTeX erstellt, wie in Abschnitt 2.2.1 angeführt. Nach Übersetzung und Konvertierung wird ein PostScript-Dokument an das Faxgerät zum Übertragen gesendet.
2. Die Angabe einer Bilddatei ist möglich.
3. Über eine einfache Webschnittstelle kann ein Faxdokument entworfen werden. Dies ist jedoch für die Zwecke der Diplomarbeit auf Grund mangelnder Formatierungsmöglichkeiten nicht geeignet.

Bei einer Koppelung diese Fax-Systems mit dem zu implementierenden elektronischen Büro könnte von diesem auch gefaxt werden, was einer weiteren Arbeitserleichterung entspräche. Variante 1 bietet einen guten Ausgangspunkt dafür.

Struktur und Inhalt

Der Aufbau des URZ-Faxdokumentes (Abbildung 2.6) ist mit keiner der vorhandenen Normen nachvollziehbar. Das Dokument trägt die große Überschrift `FAX-Mitteilung`. Die Angabe der Faxnummer ist ebenso nötig wie die Anzahl der Seiten. Ein weiterer Eintrag beinhaltet das Datum. Als fester Hinweis ist

> *„Bei Übertragungsfehlern rufen Sie bitte: (0371) 531 1551“*

vorgesehen. Neben der Anschrift des Empfängers (Firma, Abteilung, Vorname, Name, Straße und Nummer, Postleitzahl und Ort) finden sich Informationen zum Bearbeiter des Faxes.. Eine Leitwortzeile bestehend aus „`Zur Kenntnis`“, „`Zur Erledigung`“, „`Zur Stellungnahme`“ und „`Mit bestem Dank zurück`“ aus der alten FAX-Vorlage wird nicht weiter unterstützt, weil sie nie benutzt wurden. Die Felder Betreff, Anrede, Text und Gruß entsprechen denen des Briefes. Für das Fax wird nur der Kopf des Layouts für ausgehende Post (Abschnitt 2.1.5) benutzt. Den Fuß bildet eine Seitennummerierung, die sich auf allen Seiten des Dokumentes befindet.

2.2.4 Die Hausmitteilung

Mit der Hausmitteilung werden Nachrichten innerhalb der Universität versandt. Als Geschäftsabläufe kommen beide Varianten aus Abschnitt 2.1.2 in Frage. Hausmitteilungen werden ausschließlich mit LaTeX erzeugt.

LaTeX	Vorlage	buero/muster/latex/urzhausmit.tex
	Dokumentenklasse	urzhausmit.cls
	Speicherpfad	buero/hausmit/

Um den eingeschränkten LaTeX-Befehlssatz durch Einsetzen einer LaTeX-Dokumentenklasse zu verdeutlichen, werden folgend die Kommandos dargestellt, die der Textschreibende verwenden kann.

Feld für Briefkopf: Briefkopf für ausgehende Post

FAX-Mitteilung

Anzahl der Seiten:
Datum:

Bei Übertragungsfehlern rufen Sie bitte:(0371) 531 1551

Fax-Nr.:

Feld für Anschrift des Empfängers	Bearbeiter Telefon Fax

Feld für Betreff, Anrede, Text und Gruß

Seite 1

Abbildung 2.6: Struktur des Faxes

Kommando	Alias(se)	Bedeutung
\unterschrift{...}	\signature{...}	„gedruckter“ Teil der Unterschrift
\hausruf{...}	\phone{...}	4stellige Telefonnr. des Absenders
\email{...}		email-Adresse des Absenders
\uzeichen{...}		Text hinter „Unsere Zeichen“
\datum{...}		wenn anderes Datum als das aktuelle
\anrede{...}	\opening{...}	Anrede
\gruss{...}	\closing{...}	Grußformel
\von{...}		Absender, vordefinierter Wert: „URZ, 84000“
\an{...}		Empfänger (wenn nur 1 Zeile)
oder alternativ:		
\anname{...}		1. Zeile Empfänger (Name)
\anfakul{...}		2. Zeile Empfänger (Fakultät)
\anstrukt{...}		3. Zeile Empfänger (Strukturnummer)

Struktur und Inhalt

Die Hausmitteilung hat ihr eigenes Format und zählt nicht unter Geschäftsbriefe. Damit ist sie nicht an die Normen für den Geschäftsverkehr gebunden. Der Körper des Schreibens ist in Adressbereich und Textbereich aufgeteilt. Die Anschrift des Absenders ist fest und lautet

„024000, Universitätsrechenzentrum“.

024000 ist die Strukturnummer des URZ, entsprechend dem Strukturnummernverzeichnis[2] der Universität. Die Anschrift des Empfängers kann die Elemente Strukturnummer, Fakultät, Professur oder Mitarbeiter in angegebener Reihenfolge enthalten. Alle Angaben bis auf die einzelnen Mitarbeiter können dem Strukturnummernverzeichnis entnommen werden. Die neue Hausmitteilung soll nach Vereinbarung nur noch Fakultät, Professur und Name enthalten. Weiterhin können Fakultät/Institut und Dezernat/Abteilung vorkommen. Um den Absender und damit den Ansprechpartner näher zu spezifizieren, werden dessen Namen, Hausruf und e-Mailadresse angegeben. Außerdem enthält dieser Abschnitt noch den Leitsatz `Unsere Zeichen:`, wonach die Kürzel von Ansprechpartner und gegebenfalls Ersteller stehen. Das Datum befindet sich rechtsbündig über dem Textbereich. Betreffzeile, Anrede, Text und Gruß entsprechen wiederum in Struktur und Formatierungsmöglichkeiten denen des Briefes. Die Hausmitteilung nutzt das Layout für interne Post (Abschnitt 2.1.5).

2.2.5 Die Stellungnahme zur Beschaffungsanforderung

Das Universitätsrechenzentrum hat die Aufgabe, Beschaffungsanforderungen über Hardware und Software zu begutachten. Die Stellungnahme erfolgt gegenüber der Zentralen Beschaffung der TU Chemnitz. Beschaffungsanforderungen können von allen Fakultäten der Universität stammen. Allgemein bekannt ist die Stellungnahme zur Beschaffungsanforderung auch als Gutachten, welche meist von den Bearbeitern persönlich mittels LaTeX erzeugt werden.

LaTeX	Vorlage	buero/gutachten/MUSTER*.tex
	Dokumentenklasse	urzhausmit.cls
	Speicherpfad	buero/gutachten/

Struktur und Inhalt

Die Stellungnahme zur Beschaffungsanforderung (Abbildung 2.8) ist eine Hausmitteilung, die gegenüber einer normalen Hausmitteilung weitere Besonderheiten aufweist. Der Körper des Schreibens ist in einen Adressbereich und in zwei Textbereiche gegliedert. Daten über die zu begutachtende Beschaffungsanforderung sind im oberen und die sich darauf beziehende Stellungnahme

[2] Strukturnummernverzeichnis der TU Chemnitz:
http://www.tu-chemnitz.de/verwaltung/dez4/strnr/strinh.htm

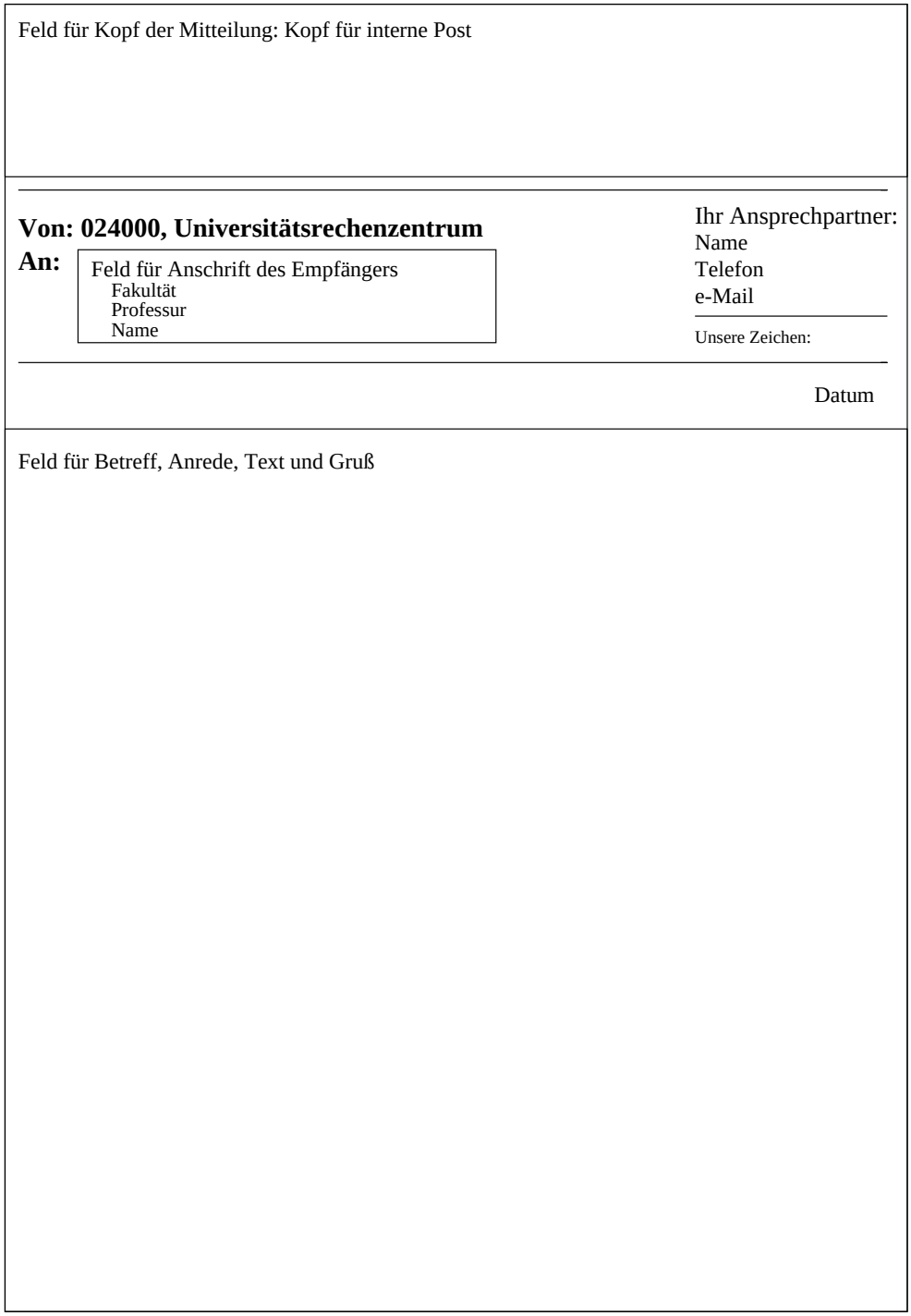


Abbildung 2.7: Struktur der Hausmitteilung

Feld für Kopf der Mitteilung: Kopf für interne Post

Von: 024000, Universitätsrechenzentrum
An: 013210
Dezernat Haushalt und Wirtschaft
Abteilung 3.2, Zentrale Beschaffung

Ihr Ansprechpartner:
Name
Telefon
e-Mail

Unsere Zeichen:

Datum

Stellungnahme zur Beschaffungsanforderung vom xx.xx.xxxx
Fakultät für ...
Professur ... / Institut für ...
Geschäftszeichen ...
(ca. xx,- DM)

Feld für die Stellungnahme

Prof. Dr.-Ing. habil. Hübner
URZ-Leiter

Abbildung 2.8: Struktur des Gutachtens

im unteren Bereich zu finden. Wie auch in der einfachen Hausmitteilung nennt der Absender die Struktureinheit des Universitätsrechenzentrums. Die Anschrift des Empfängers ist konstant und lautet

„013210, Dezernat Haushalt und Wirtschaft, Abteilung 3.2, Zentrale Beschaffung“.

Außer der Angabe nur eines Kürzels nach „`Unsere Zeichen:`“, werden die Daten des Ansprechpartners wie in der Hausmitteilung spezifiziert. Im Fall des Gutachtens ist der Ersteller des Dokumentes mit dem Ansprechpartner gleichzusetzen. In fetter Schrift und zentriert angeordnet folgen im ersten Textbereich:

„Stellungnahme zur Beschaffungsanforderung vom [Datum]“
„Fakultät für [Name der Fakultät]“
„Professur [Name der Professur]“
„Geschäftszeichen [...]“
„(ca. [Preis],- DM)“.

Anstelle der der Fakultät kann auch die Zentrale Einrichtung und anstelle der Professur ein Dezernat oder Institut aufgeführt werden. Für die Stellungnahme stehen folgende Varianten zur Auswahl, die auch durch zusätzlichen Text ergänzt werden können.

1. *Gegen eine Beschaffung gibt es seitens des URZ keine fachlichen Einwände.*
2. *Gegen eine Beschaffung der Software gibt es seitens des URZ keine fachlichen Einwände.*
3. *Gegen eine Beschaffung der Software [Bezeichnung] gibt es seitens des URZ keine fachlichen Einwände.*
4. *Der Beschaffungsanforderung liegt KEIN Angebot bei.*
5. *Der Beschaffungsanforderung liegt nur ein Angebot bei.*
6. *Das URZ kann keine Unterstützung bei der Anwendung der o.g. Software gewährleisten.*
7. *Für diese Systeme kann durch das URZ keine technische Unterstützung zugesichert werden. Grund dafür sind die teilweise fehlenden technischen Angaben zu den eingesetzten Hardwarekomponenten.*
8. *Bei der Beschaffung handelt es sich um Spezialkomponenten, diese Technik unterliegt nicht der Begutachtungspflicht durch das URZ.*
9. *Die Finanzierung erfolgt durch Mittel der Struktureinheit.*

Die festen Unterschriftszeilen

„Prof. Dr.-Ing. habil. Hübner
URZ-Leiter“

werden meist im Auftrag unterzeichnet. Beim Gutachten wird das Layout für interne Post (Abschnitt 2.1.5) verwendet.

2.2.6 Die Zugangsberechtigung

Die Zugangsberechtigung hat die Aufgabe, einer Person den Zugang zu Diensträumen der TU Chemnitz außerhalb der Regelarbeitszeit zu erlauben. Bei den Zugangsberechtigten handelt es sich vor allem um die Mitarbeiter des Universitätsrechenzentrums. Dazu wird Studenten in einzelnen Fällen Zugang erteilt. Die Berechtigung wird üblicherweise vom Personal der Sekretariats für ein gesamtes Jahr beantragt. Am Ende eines Jahres werden für alle URZ-Mitarbeiter, die z.B. wegen Havarie- oder Installationsarbeiten den Zugang zu Arbeitsräumen außerhalb der Regelarbeitszeit benötigen, Zugangsberechtigungen für das nächste Jahr erstellt. Die Schreiben werden zu einem vereinbarten Termin erstellt und danach die Unterschriften in folgender Reihenfolge eingeholt.

1. Mitarbeiter
2. stellvertretender Leiter des URZ
3. Leiter des URZ
4. Betriebssicherheit

Wenn für einen neuen Mitarbeiter oder Studenten Zugangsberechtigungen während des Jahres angefertigt werden, bleibt das Verfahren gleich.

Zugangsberechtigungen werden in der Regel mit StarOffice erstellt und nur ausgedruckt, aber nicht gespeichert. Unter LaTeX gibt es allerdings auch Vorlagen dazu.

LaTeX	Vorlagen	buero/muster/latex/zugangsberechtigung-mit.tex
	Dokumentenkl.	article.cls (Standard)
StarOffice	Vorlage	buero/muster/StarOffice/zugangsberechtigung.vor

Struktur und Inhalt

Die Zugangsberechtigung unterliegt keiner Norm und besteht aus einer zentrierten Überschrift, einem Text- und einem Unterschriftsbereich. Im Textbereich sind Datum, Anrede und Name des Zugangsberechtigten und Gültigkeitsdauer der Berechtigung variabel. Der Text zur Berechtigung ist fest und lautet wie in Abbbildung 2.9 dargestellt. Im Unterschriftsfeld sind die Unterschriften von Herrn Prof. Dr. Uwe Hübner (Leiter URZ), Herrn Detlef Lämmel (Fachgebietsleiter Betriebssicherheit), Herrn Lothar Kempe (stellvertretender Leiter des URZ) und des Zugangsberechtigten vorgesehen. Die Zugangsberechtigung ist zwar kein Dokument, das außerhalb der Universität benötigt wird, trägt aber trotzdem das Layout für die ausgehende Post.

Zur vereinfachten Generierung von Zugangsberechtigungen für die URZ-Mitarbeiter könnte eine Serienbrieffunktionalität vorgesehen werden.

2.2.7 Der Beschaffungsantrag

Der Beschaffungsantrag wird an die Zentrale Beschaffung der TU Chemnitz gesandt, um einen Beschaffungsvorgang auszulösen. Die Zentrale Beschaffung erstellt einen Kaufvertrag anhand der Angaben im Beschaffungsantrag. Das Schreiben wird nach Variante I aus 2.1.2 im Sekretariat unter strenger Einhaltung der Unterschiftenberechtigung erstellt. Es hat seinen Ursprung in einem Papierformular, das mit der Schreibmaschine ausgefüllt werden muss. Der URZ-Mitarbeiter und gleichwohl Empfänger der Lieferung stellt dem Sekretariat die notwendigen Daten des Beschaffungsantrages zur Verfügung. Beschaffungen werden nur mit StarOffice erzeugt.

StarOffice	Vorlage	buero/muster/StarOffice/beschaffung-*.vor
	Speicherpfad	buero/beschaffungen/

Struktur und Inhalt

Der Beschaffungsantrag umfaßt zwei Seiten (Abbildung 2.10 und Abbildung 2.11), die in mehrere aufeinanderfolgende Teile gegliedert werden können. In einem Adressbereich werden Bedarfsstelle und Anschrift des Empfängers und Absenders aufgeführt. Der Bedarfsstelle *„Universitätrechenzentrum“* folgt die Anschrift des Empfängers und lautet:

„Beschaffungsantrag an den Kanzler der TU Chemnitz“
„-Zentrale Beschaffung-“.

In der Anschrift des Absenders sind

- `Ort,Datum` *(Chemnitz)*,

Feld für Kopf: Briefkopf für ausgehende Post

Zugangsberechtigung

zu Diensträumen der TU Chemnitz außerhalb der Regelarbeitszeit

Chemnitz,den ...

Herr **Name,**

Mitarbeiter des Universitätsrechenzentrums der TU Chemnitz ist berechtigt, außerhalb der Regelarbeitszeit sowie sonnabends, sonn- und feiertags sich in den Arbeitsräumen aufzuhalten, in denen er seine notwendigen Arbeitsaufgaben ausführt.

Der Mitarbeiter ist darauf hingewiesen worden, dass nur solche Arbeiten allein ausgeführt werden dürfen, bei denen keine Gefährdungen auftreten. Sind Gefährdungen zu erwarten, ist in jedem Fall dei Anwesenheit eines weiteren Mitarbeiters erforderlich.

Gültig: **01.01.20xx - 31.12.20xx**

Prof. Dr. Uwe Hübner
Leiter URZ

Detlef Lämmel
Fachgebietsleiter Betriebssicherheit

Zugangsberechtigter

Lothar Kempe
stellv. Leiter URZ

Feld für Fuss: Brieffuß für ausgehende Post

Abbildung 2.9: Struktur der Zugangsberechtigung

Bedarfsstelle

Universitätsrechenzentrum

Beschaffungsantrag an
den Kanzler der

TU Chemnitz
-Zentrale Beschaffung-

Ort, Datum

Anschrift

Zuständige Bearbeiter

Fernsprecher

Geschäftszeichen
024000/00xxx/xxx99-0x

Nachfolgend aufgeführte Lieferungen/Leistungen werden beantragt:
(weiter Angaben - soweit erforderlich - auf gesondertem Blatt)

lfd. Nr.	Bezeichung	Menge	Preis in DM je Einheit	Preis in DM Gesamt
01				
02				
03				
04				
05				
06				
07				
		16,00%	Mwst.	
			Summe	

Empfänger und Bestimmungsort:
Feld für Anschrift des Mitarbeiters

vorgeschlagene Lieferfirma:
Feld für Anschrift der Lieferfirma, inkl. Tel.- und Faxnummer

Einbauteile, Zusatzgeräte nein
inv-Nr. Bezeichnung des Hauptgerätes inv-Nr. Bezeichnung des Hauptgerätes
ja, zu

Liefertermin/Lieferfrist
baldmöglichst

Ersatzteile, keine Inventarisierung!

Abbildung 2.10: Struktur des Beschaffungsantrages Seite 1

Bedarfsbegründung
Notwendigkeit der Maßnahme

Angaben zu personellen und sächlichen Folgekosten

Angaben zu Räumlichkeiten und Installationen

Sonstige Angaben:
Vergleichsangebote:

Es wird bestätigt, dass die angeforderten Gegenstände bzw. Leistungen zur Erfüllung der Aufgaben der Bedarfsstelle unter Beachtung des Grundsatzes der Sparsamkeit und Wirtschaftlichkeit zum betragten Zeitpunkt erforderlich sind. Die zweckentsprechende Verwendung ist gesichert. Es ist geprüft, dass der ermittelte Bedarf aus den vorhandenen Beständen nicht gedeckt werden kann bzw. die Möglichkeit der Ausleihe/Mitbenutzung nicht besteht.

Unterschrift einer berechtigten Bediensteten

- von der mittelbewirtschaftenden Stelle auszufüllen -

Geschäftszeichen
024000/00xxx/xxx99-0x

Die zur Durchführung der Maßnahmen erforderlichen Haushaltsmittel stehen

	Betrag in DM	Kapitel	Titel	VF-Thema
zur Verfügung und sind vorgemerkt				

nicht zur Verfügung. Die Bedarfsstelle ist zu informieren.

(Datum, Unterschrift des zuständigen Bediensteten)

(Datum, Unterschrift des BdH, soweit Beteiligung erforderlich)

Abbildung 2.11: Struktur des Beschaffungsantrages Seite 2

- `Anschrift` *(Universitätsrechenzentrum)*,
- `Zuständige Bearbeiter` *(Personal des Sekretariat)*,
- `Fernsprecher` *(0371/531-1551 und -1716)* und
- `Geschäftszeichen`

anzugeben. Das Geschäftszeichen ist wie folgt aufgebaut: 024000/JJxxx/ttttt-uu. 024000 bezeichnet die Struktur und entspricht der Kostenstelle. JJ steht für die beiden letzten Ziffern der Jahreszahl, xxx ist eine laufende Nummerierung. ttttt bezeichnet den Titel und uu den Untertitel. Die beantragten Lieferungen oder Leistungen werden in einer Tabelle dargestellt, deren einzelne Felder

- `lfd. Nr.`,
- `Genaue Bezeichnung des Gegenstandes bzw. der Leistung (ggf. Bestell-Nr. des vorgeschlagenen Lieferanten)`,
- `benötigte Menge`,
- *veranschlagter Preis in DM* `je Einheit` und
- `Gesamt`

lauten. Unter sieben vorgesehenen Positionen wird die Mehrwertsteuer vermerkt, die entweder sieben oder 16 Prozent betragen kann. Die darauffolgende Zeile in der Tabelle enthält die Summe, die sich aus aus den Beträgen der Gesamtpreise pro Position und der Mehrwertsteuer ergibt. Unter dieser Tabelle werden Empfänger, Bestimmungsort und eine vorgeschlagene Lieferfirma angegeben. Die Empfängerzeile enthält die Anschrift eines URZ-Mitarbeiters, bestehend aus Name, Straße und Zimmer. Im Feld der Lieferfirma werden Angaben zu Firmenname, Straße, Postleitzahl, Ort, Telefon- und Faxnummer getätigt. Der letzte Teil der ersten Seite besteht aus den zwei fest eingestellten Einträgen *„Einbauteile, Zusatzgeräte“* - `nein` und *„Liefertermin/Lieferfrist“* - `baldmöglichst`. Weiterhin ist eine der Optionen *„Ersatzteile, keine Inventarisierung!“* und *„Inventarisierung, Hinweis siehe Rückseite“* zu wählen.

Die zweite Seite beginnt mit einem Textfeld Bedarfbegründung, in dem die Notwendigkeit der Maßnahme erläutert werden soll. Zwei weitere Textfelder *„Angaben zu personellen und sächlichen Folgekosten“* und *„Angaben zu Räumlichkeiten und Installationen“* enthalten den festen Eintrag *„entfällt“*. Im darauffolgenden letzten Textfeld *„Sonstige Angaben“* können Vergleichsangebote dargelegt werden. Die Textbereiche enthalten Standardschrift ohne weitere Formatierung. Der zur nachfolgenden Unterschrift berechtigte Bedienstete entspricht dem Empfänger von der ersten Seite der Beschaffungsanforderung und signiert das Dokument als Erster. Der letzte Teil ist von der mittelbewirtschaftenden Stelle auszufüllen. Trotzdem sind neben dem selben Geschäftszeichen wie auf der ersten Seite

- *der Betrag in DM, (Summe von der ersten Seite)*
- *Kapitel (entweder 1210 oder 1207) und*
- *Titel (Ausschnitt aus dem Geschäftszeichen)*

aufzuführen. (*VF-Thema* bleibt frei) Eine weitere Unterschrift wird am Ende der zweiten Seite nach dem Datum von einem zuständigen Bediensteten getätigt, welcher der Leiter der Struktureinheit oder eine delegierte Person ist. Sie wird nur geleistet, wenn die oben erwähnte Unterschrift des berechtigten Bediensteten vorhanden ist. Der Beschaffungsantrag ist ein Formular, verwendet kein Layout im Sinne von Kopf und Fuß und umfasst zwei Seiten ohne Nummerierung.

Im letzten Monat der Diplomarbeit wurde ein neues Formular für den Beschaffungsantrag eingeführt, welcher aus zeitlichen Gründen nach Absprache mit den Betreuern nicht mehr eingearbeitet werden konnte.

2.2.8 Bilden von Dokumentenklassen

Wie in der Analyse der Dokumententypen angedeutet, enthalten viele Typen die selben Elemente oder ähnliche Strukturen. Somit können die Dokumententypen klassifiziert werden, was die spätere Weiterverarbeitung der erzeugten Dokumente vereinfachen soll und eine bessere Übersicht über die Dokumentenarten bietet. Alle analysierten Dokumententypen lassen lassen sich nach ihrer Struktur in einem hierarchischen Schema darstellen, in dem sich der Verwandtschaftsgrad leicht erkennen lässt (Abbildung 2.12).

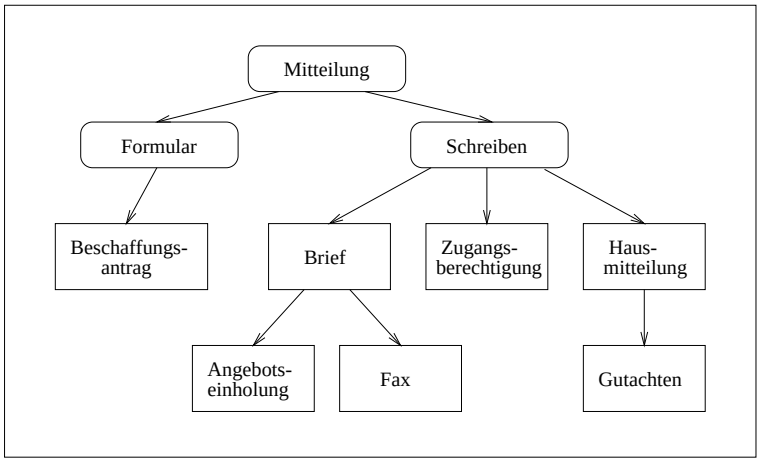


Abbildung 2.12: Verwandtschaft der Geschäftsdokumente

Einzelne Dokumententypen lassen sich von anderen ableiten, weshalb auch von Klassen gesprochen wird. Alle Dokumentenarten werden als Mitteilungen zum Informationsaustausch betrachtet, die sich in Formulare und Schreiben unterteilen. Ein Schreiben ist eine Mitteilung, deren Inhalt vom Nutzer entworfen und strukturiert werden kann, wogegen sich ein Formular an eine vorgegebene Struktur hält.

Angebotseinholung und Fax ähneln in der Struktur einem Brief und enthalten beide weitere Elemente, z.B. im Kontaktinformationsbereich. Die Angebotseinholung beinhaltet zusätzlich eine Artikelliste und bei Faxen ist die Faxnummer des Empfängers von Bedeutung. Da das Gutachten im Grunde eine Hausmitteilung ist, wurde schon in der Analyse erläutert. Deshalb ist es teilweise wie eine solche aufgebaut, enthält aber weitere Bestandteile. Die Zugangsberechtigung, die auch zu der Mitteilung zählt, findet unter den anderen Dokumententypen keine Entsprechung. Ebenso und schon auf Grund der Zugehörigkeit zu den Formularen bildet der Beschaffungsantrag eine eigene Klasse. Somit konnten vier Dokumentenklassen erstellt werden, die mit dem Namen des Dokumententyps bezeichnet wurden, von dem weitere Dokumentenarten abgeleitet wurden.

1. Brief
2. Hausmitteilung
3. Zugangsberechtigung
4. Beschaffungsantrag

2.3 Zusammenfassung

Die derzeitige Erstellung von Geschäftsdokumenten kann durch das Einführen eines modernen Systems rationalisiert werden. Aus der Einleitung war zu erfahren, was Digitalisierung der

Geschäftsprozesse bedeutet. Nachfolgend werden durch die Analyse erkannte Faktoren der derzeitigen Geschäftsvorfälle zusammengefasst und Ideen für eine effektivere Gestaltung dieser vorgestellt.

Ohne die Mächtigkeit und Flexibilität von LaTeX in Frage zu stellen, gestaltet sich die Erstellung von Geschäftsdokumenten ziemlich schwerfällig. Mit der Einführung von StarOffice ergaben sich einige Erleichterungen für die Nutzer. Allerdings brachte die parallele Verwendung zweier unterschiedlicher Systeme wiederum Probleme mit sich. Ein Web-basiertes System läuft unabhängig vom Betriebssystem und ist im gesamten Intranet verfügbar, was auch keine Installation einer neuen Textverarbeitungssoftware auf jedem Nutzerrechner erfordert. In einer modernen Infrastruktur können Ausfüllhilfen und weitere Erleichterungen zur Dokumentenerzeugung integriert sein, wodurch Präzision und Erstellungsgeschwindigkeit erhöht werden. Auch in elektronischen Geschäftsprozessen wird das Papier weiterhin eine entscheidende Rolle spielen. Wie die Gartner Group analysierte, werden e-Business-Aktivitäten das Drucken eher fördern als eliminieren. [Gar00] Manchmal werden Papiernachweise benötigt und in vielen Fällen fordert das Gesetz handschriftliche Unterschriften. Das Aushängeschild jedes Unternehmen ist ein einheitliches Layout der Geschäftsdokumente, die Corporate Identity, welche sich mit geeigneter Software zur Dokumentenerstellung realisieren lässt. Einzuhaltende Normen und ein definiertes Maß an Gestaltungsmöglichkeiten zur Textformatierung können durchgesetzt werden. Durch das Einführen von Berechtigungen kann der Zugriff auf Dokumente für bestimmte Personen oder Nutzergruppen beschränkt werden, was derzeit nicht der Fall ist. Die Einrichtung eines zentralen Systems vermindert den Administrationsaufwand und gewährleistet das korrekte Sichern von Dokumenten in vorgesehenen Verzeichnissen und als einer Konvention unterliegenden Namen. Eingescannte Unterschriften werden auf Grund auftretender Sicherheitsmängel nicht unterstützt. Um Integrität und Authentizität der Dokumente nachweisen zu können, ist die Anwendung einer digitalen Signatur eine geeignete Lösung. Weitere Sicherheitskonzepte, wie etwa Verschlüsselung, sind nicht nötig. Durch den Einsatz eines einfachen und bequemen Web-Interfaces sollten die Mitarbeiter des Universitätsrechenzentrum in die Lage versetzt werden, ihre Dokumente persönlich schneller schreiben zu können als sie erst nach einer Übermittlung der Wünsche durch eine weitere Person anfertigen zu lassen. Einige der analysierten Dokumente gleichen sich in ihren Strukturen, sodass Dokumentenklassen gebildet wurden, die die spätere Implementation erleichtern könnten.

Kapitel 3

Die Extensible Markup Language

Nach der Entwicklung von XML konnte eine vermehrte Konvertierung von Daten aus den unterschiedlichsten Bereichen und einer Vielzahl von Formaten nach XML beobachtet werden, die bis heute anhält. Binäre Dokumentendaten werden lesbarer Text, die jeder Mensch lesen und schreiben kann. XML verändert nicht nur die Denkweise über die Daten, sondern auch über das Web. Tim Bray, einer der Schöpfer von XML und Autor der XML-Spezifikation meinte sogar: „Als ich mich dem Projekt anschloss, ahnte ich noch nicht, dass man damit die Welt beherrschen kann." [Sch99] Um die Grundlagen und Konzepte dieser Sprache zu verdeutlichen, werden sie und mit XML in naher Verbindung stehende und für die weitere Arbeit entscheidende Bereiche in diesem Kapitel angesprochen. Weiterhin soll geklärt werden, wie diese Technologie als Basis für eine moderne Infrastruktur zur Erledigung von Geschäftsvorfällen eingesetzt werden kann.

3.1 Einführung in XML

Die Einführung bietet einen sehr kurzen Überblick über das ständig wachsende Feld der Extensible Markup Language und ihrer Teilbereiche.

3.1.1 Entstehung von XML

XML ist ein Projekt des WWW-Konsortiums (W3C) und wurde seit Juli 1996 von einer Arbeitsgruppe unter Aufsicht des W3C Editorial Review Boards entwickelt. [Mic99] Unter dem Vorsitz von Jon Bosak (Sun Microsystems) waren daran SGML- und Internet-Experten namhafter Softwarefirmen wie Microsoft, Novell und IBM beteiligt. Hintergrund der Initiative war zum einen die ungewisse Zukunft von HTML, welcher herstellerspezifische Erweiterungen durch Software-Anbietern bevorstand. Zum anderen war der als möglich erachtete Einsatz von SGML als technologische Alternative im Intra-/Internet mit unakzeptablen Aufwänden für Implementierung und Betrieb verbunden. Der Nachteil der extremen Komplexität der schwach verbreiteten SGML steht Vorteilen wie Plattformunabhängigkeit, hohe Flexibilität und strukturierte Erzeugung und Verarbeitung von Dokumenten gegenüber. Bei Beibehaltung der Vorteile von SGML [SGML] sollte die neue Sprache aber weniger komplex sein und eine einfache Nutzbarkeit über das Internet bieten. XML ist ein Kompromiss zwischen der Einfachheit von HTML und Funktionsvielfalt von SGML. Sie garantiert maximale Flexibilität in der Definition von Datenformaten bei vertretbarem Aufwand. XML wurde im November 1996 als Entwurf vorgestellt und hat seit Februar 1998 als XML 1.0 [XML] den Status einer W3C-Empfehlung.

Im Internet gibt es viele Möglichkeiten sich über XML und die verwandten Sprachen, über verfügbare Software, neuste Entwicklungen und Publikationen aktuell zu informieren. Die Oasis-Site von Robin Cover[1] ist dafür zum Beispiel ein geeigneter Ausgangspunkt.

[1] http://www.oasis-open.org/cover/xml.html

3.1.2 Was ist XML?

XML war ursprünglich als HTML-Alternative und eine auf Internet-/Intranet-Publikationen zielende Sprache gedacht. [Mic99] Auf Grund ähnlicher Mächtigkeit, aber besserer Handhabbarkeit gegenüber SGML ist sie als eine Sprache zum Austausch, Wiederfinden und zur Verwaltung semantisch spezifizierter Daten geeignet. XML garantiert Kompatibilität von Daten in beliebigen Verwendungszusammenhängen und ermöglicht intelligente inhaltsbezogene Recherche. Derzeit liegt der Schwerpunkt des Einsatzes im e-Business beim Online-Austausch von Informationen.

Metasprache

XML ist eine formale Sprache zur Definition von Auszeichnungssprachen beliebigen Umfangs für viele unterschiedliche Zwecke. Darum wird XML als Metasprache bezeichnet, die unspezifisch hinsichtlich der Art der auszuzeichnenden Dokumente ist. Die Objekte von XML sind konkrete Auszeichnungssprachen (Abbildung 3.1), die das Vokabular zur Auszeichnung von Dokumenteninhalten enthalten.

Abbildung 3.1: Objekte von XML

XML-Applikationen

Die mit XML definierten Auszeichnungssprachen nennt man auch XML-Applikationen. Bei diesen Sprachen handelt es sich nicht um Programme, die Dokumente verarbeiten, sondern um Sprachen, die die Struktur von Dokumenten normieren und der Beschreibung von Dokumentinhalten dienen. Die Verwendung der mit XML-Anwendungen erstellten Beschreibungen und die Art der Verarbeitung werden von XML und der Auszeichnungssprache nur teilweise vorbestimmt. Abbildung 3.2 verschafft einen kleinen Überblick über existierende Auszeichnungssprachen.

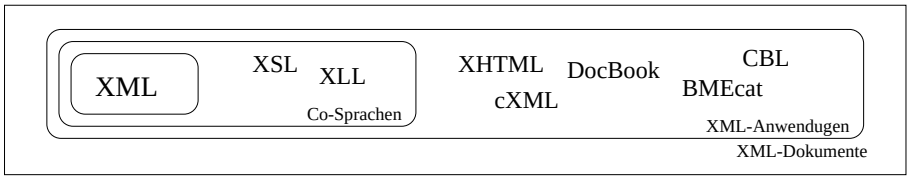


Abbildung 3.2: Einordnung von Auszeichnungssprachen

3.1.3 Vorteile der Verwendung von XML

Ein bedeutender Vorteil von XML ist die Austauschbarkeit der Dokumente. XML trennt Layout und Inhalt von Dokumenten und erlaubt eine portable Darstellung des Inhaltes. Zugleich wird der Inhalt so gekennzeichnet, dass ihm jederzeit eine beliebige Formatierung zugewiesen werden kann. Damit werden XML-Dokumente von der Systemplattform, bestimmten Anwendungsprogrammen und den Medien ihrer Präsentation unabhängig. Der Nachteil besteht im Aufwand für die Nutzung spezieller Software und Layouts zur Verarbeitung von XML-Dokumenten. Dagegen steht allerdings die erhebliche Ersparnis an Konvertierung und Integration bei Plattform-, Applikations- oder Medienwechseln.

XML überlagert die Inhalte von Dokumenten mit Metainformationen, die die Rolle von Dokumentteilen beschreibt und dabei Informationseinheiten hierarchisch identifiziert. Mit Hilfe der semantischen Tags können Informationseinheiten gezielt angesteuert, kopiert oder aktualisiert werden. Damit wird der Zugang zu den Daten optimiert, deren Pflege vereinfacht und Redundanz durch Mehrfachnutzung vermieden. Diese Vorzüge einer inhaltlichen Modularisierung müssen durch entsprechende Software unterstützt werden.

Weitere Vorteile betreffen die inhaltliche und formale Qualität von Dokumenten. Mit XML können Strukturen definiert werden, die ein Modell für den gedanklichen Aufbau eines Dokumententyps darstellen. Damit verringert sich auch die Gefahr, dass wesentliche Teile fehlen, andere mehrfach erscheinen und wieder andere falsch platziert sind. Die Formatierung wird vom Inhalt getrennt und ist so gut, wie die darauf bezogenen Stylesheets. Durch die inhaltliche Auszeichnung wird die inhaltliche und formale Qualität des Dokumentes während des Erstellungsprozess gewahrt.

Austauschbarkeit, Modularisierbarkeit und Qualitätssicherung sind die wesentlichen Nutzenaspekte, die XML mit SGML teilt. Der deutlich verringerte Sprachumfang erleichtert nicht nur die Implementierung, sondern beschleunigt auch die Verarbeitung von XML-Dokumenten. Auf Grund der erwähnten Vorteile eignet sich XML auch besser für den Einsatz im Internet als SGML. Zu den Vorteilen, die man sich hier von XML verspricht, zählen insbesondere:

1. Auf Grund der durch XML eröffneten Möglichkeiten seiner semantischen Auszeichnung von Dokumenten können sie durchsucht werden, was besonders im Internet von großer Bedeutung ist.

2. Dokumente können personalisiert werden. Der Empfänger bekommt die Information, die er zu brauchen glaubt.

3. Die Aufbereitung von Informationen für ihre Präsentation kann Klientenrechnern oder zwischengeschalteten Server übernommen werden. Übertragen wird nur die angeforderte Nettoinformation, was die Server entlastet und die Durchsatzproblematik entschärft.

[Mic99]

3.1.4 Syntax

Die XML-Markierungen in den Dokumenten (die XML-„Tags“) bezeichnen die inhaltlichen (strukturellen) Einheiten von Dokumenten. Vokabular und Syntax der Auszeichnungssprachen sind frei gewählt.

Das folgende Beispiel zeig ein XML-Dokument mit einer einfachen Briefstruktur, in dem der Absender mit Namen und Ort spezifiziert ist. Die Nachricht des Briefes ist zwischen den Tags `<message>` und `</message>` enthalten.

```
<?xml version="1.0" standalone="yes" ?>
<letter id="1001" date="2000-06-21">
  <from>
    <name> Rico </name>
    <city> Chemnitz </city>
  </from>
  <message>
    Das ist ein <strong>XML</strong>-Dokument.
  </message>
</letter>
```

An dem Beispiel lassen sich schon einige strukturellen Merkmale von XML erkennen, z.B. die Pflicht, jedes geöffnete Tag wieder schließen zu müssen. Genaue Informationen zur Syntax und den damit verbundenen Techniken (Entitäten, Namensräume, usw.) können im Internet und in der einschlägigen Literatur nachgelesen werden. [Bra98], [Gol99]

3.2 Dokumententypen

XML dient der Beschreibung von typischen Inhalten und ihren Beziehungen, zu denen man durch Abstraktion ähnlicher Dokumente gelangt. In einem Brief geht es beispielsweise nicht um eine besondere Anschrift, Anrede oder speziellen Text, sondern darum, dass jeder Brief eine Anschrift, eine Anrede und einen Text hat oder haben kann und dass die Reihenfolge der Inhalte Anschrift, Anrede, Text ist oder sein kann.

Es geht also um Inhaltstypen für einen bestimmten Dokumententyp (z.B. einen Brief) und dessen Beziehungen. Diese Inhaltstypen kommen nicht nur in diesem oder jenem Dokument vor, sondern sind charakteristisch für alle Dokumente eines bestimmten Typs. Man spricht darum in XML von Dokumenttypen als Spezifikation von Inhaltstypen und ihren Beziehungen.

Mit XML läßt sich ein solcher Dokumenttyp definieren, und zwar in zwei Varianten auf unterschiedliche Weise. Hierbei handelt es sich um die Spezifikation mittels Dokumenttyp-Definition (DTD) oder mittels XML-Schema. Welche Inhaltstypen es in Dokumenten eines bestimmten Typs geben kann oder muss und in welchen Beziehungen diese Inhaltstypen stehen können oder müssen, wird durch XML festgelegt. Eine solche Spezifikation entspricht der Definition einer Auszeichnungssprache.

3.2.1 Dokumenttyp-Definition

Die Document Type Definition (DTD) schreibt vor, wie ein Dokument des betreffenden Typs auszuzeichnen ist. Zur Spezifikation sind Deklarationsregeln zu beachten und eine spezielle Syntax zu verwenden.

Im folgenden Beispiel ist das obige XML-Dokument des Briefes um eine DTD erweitert. In der Dokumenttyp-Definition ist vorgeschrieben, dass sich die Tags `from` und `message` innerhalb des Elementes `letter` befinden müssen, welches die obligatorischen Attribute `id` und `date` besitzt. `#PCDATA` steht für „parsed character data“ und entspricht einfachem Text.

```
<?xml version="1.0" ?>
<!DOCTYPE letter [
<!ELEMENT letter  (from, message)>
<!ATTLIST letter  id   ID    #REQUIRED
                  date CDATA #REQUIRED>
<!ELEMENT from    (name, city)>
<!ELEMENT name    (#PCDATA)>
<!ELEMENT city    (#PCDATA)>
<!ELEMENT message (#PCDATA | strong)*>
<!ELEMENT strong  (#PCDATA)>
]>
<letter id="1001" date="2000-06-21">
  <from>
    <name> Rico </name>
    <city> Chemnitz </city>
  </from>
  <message>
    Das ist ein <strong>XML</strong>-Dokument mit DTD.
  </message>
</letter>
```

Die Dokumenttyp-Definition kann getrennt zum Dokument stehen oder wie im Beispiel in ihm enthalten sein. Zu weiterführenden Informationen sind die in Kapitel 3.1.4 aufgeführten Quellen ratsam.

Gültigkeit und Wohlgeformtheit

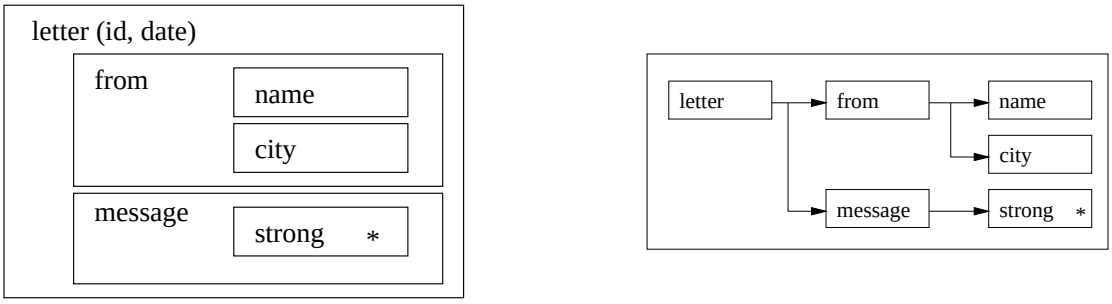
Wenn ein Dokument im Sinne seiner Dokumenttyp-Definition korrekt ausgezeichnet wurde, nennt man es relativ zur DTD gültig oder valid. Ein valides Dokument bezeichnet man als Instanz eines Dokumenttyps. Eine Dokumenttyp-Definition definiert somit eine Klasse von Dokumenten. Jedes

Dokument ist Element dieser Klasse und damit Instanz der DTD, wenn es dieser entsprechend ausgezeichnet wurde.

Ein validierender Parser prüft die Instanz gegen die Regeln der Dokumenttyp-Definition. Beim Lesen der Dokumenttyp-Definition und der Instanz bemerkt und meldet er Verstöße gegen die Auszeichnungsregeln und gegen die Deklarationsregeln von XML. Ein nicht-validierender Parser beschränkt sich auf die Prüfung der Instanz gegen die Auszeichnungsregeln von XML. Er entscheidet lediglich, ob ein Dokument im Sinne von XML well-formed ist oder nicht. Wohlgeformt bedeutet dabei, dass das Markup eines Dokument der XML-Syntax entspricht. Zur Wohlgeformtheit von XML-Dokumenten gehört zum Beispiel, dass geöffnete Tags geschlossen und geschachtelte Tags in rückwärts geschachtelter Reihenfolge geschlossen werden müssen. Auch Klein- und Großschreibung wird unterschieden.

Grafische Darstellung

Sowohl Dokumenttyp-Definitionen als auch entsprechend ausgezeichnete Dokumente können zur besseren Überschaubarkeit grafisch dargestellt werden. Eine solche Darstellung ist eine wichtige Hilfe bei der Analyse, beim Design und nicht zuletzt bei der Diskussion über Dokumentstrukturen. Jeder Codierung von Dokumenttyp-Definitionen sollte ihre grafische Darstellung vorangehen, die eine bessere Übersicht über die Struktur einer Instanz schafft und in vielen XML-Editoren implementiert ist. Gebräuchliche Darstellungsmittel sind das Baumdiagramm, das Flussdiagramm und das Schachteldiagramm.



(a) Schachteldiagramm (b) Baumdiagramm

Abbildung 3.3: Schachtel- und Baumdiagramm eines XML-Dokumentes

Die Abbildung 3.3 stellt die obige Dokumenttyp-Definition in Schachtel- und Baumnotation dar. Die Darstellungen der Dokumententypen aus der Analyse von Abschnitt 2.2 lassen Gemeinsamkeiten mit dem Schachteldiagramm erkennen, die in der weiteren Arbeit nützlich sein werden.

3.2.2 XML-Schema

XML-Schemata [XML-Schema] definieren wie DTDs Dokumenttypen und formulieren Regeln für die Auszeichnung und damit für die Struktur und Inhalt von Dokumenten. Diese Regeln beziehen sich nicht anders als die in DTDs spezifizierten Regeln auf Elemente, Attribute und Entitäten und auf die strukturellen Beziehungen zwischen den Elementen. Was Schemata und DTDs unterscheidet, ist zumindest vordergründig nur ihre Syntax. In DTDs kommt eine Deklarationssyntax zum Einsatz, die sich deutlich von der Auszeichnungssyntax von XML-Dokumenten unterscheidet. Die in Schemata verwendete Deklarationssyntax dient auch zur Auszeichnung von Dokumenten.

In folgendem Beispiel ist die DTD aus Abschnitt 3.2.1 zur Veranschaulichung als Schema dargestellt.

```
<?xml version="1.0" ?>
<!DOCTYPE schema SYSTEM "structures.dtd">
```

```
<schema>
  <element name="letter">
    <type>
      <element ref="from"/>
      <element ref="message"/>
      <attribute name="id"   type="ID"    minOccurs="1"/>
      <attribute name="date" type="date"  minOccurs="1"/>
    </type>
  </element>
  <element name="from">
    <type>
      <element ref="name"/>
      <element ref="city"/>
    </type>
  </element>
  <element name="name" type="string"/>
  <element name="city" type="string"/>
  <element name="message" type="string">
    <type>
      <element ref="strong" minOccurs="0" maxOccurs="*"/>
    </type>
  </element>
  <element name="strong" type="string"/>
</schema>
```

An diesem Beispiel, das dem Working Draft des XML-Schema [XML-Schema] vom 7. April 2000 entspricht, lässt sich schon die Mächtigkeit von Schemata erkennen. Mittlerweile liegt das XML-Schema als weiterentwickelte Empfehlung[2] vor, die einige Unterschiede zum Working Draft aufweist.

Syntaktisch gibt es keine Unterscheidung zwischen Deklaration und Verwendung von Datenobjekten. Während bestimmten Elementtypen und Attribute ein Schema definieren, zeichnen andere Elementtypen und Attribute ein Dokument aus. Die Elementtypen und Attribute, die ein Schema definieren sollen, müssen in einer Schema-Sprache spezifiziert sein. Mit welchen Mitteln diese Schema-Sprache wiederum spezifiziert ist, spielt keine entscheidenden Rolle. Schema-Sprachen müssen nicht durch Deklarationssyntax von XML definiert werden und treten vielmehr an die Stelle der Deklarationsregeln von XML.

Entwicklung von XML-Schema

XML-Schema wurde von der Arbeitsgruppe XML-Schema Working Group des W3C entwickelt. Dies geschah unter Berücksichtigung von öffentlich diskutierten Vorschlägen für Schema-Sprachen, die nach Bekanntgabe der Einführung und Voraussetzungen von Schemata durch das W3C von verschiedenen Firmen eingereicht worden. Der Working Draft des W3C leitet sich unter anderem aus vier Vorschlägen (XML-Data, DCD, SOX, DDML) ab und besteht aus zwei Teilen.

1. *XML Schema Part 1: Structures* spezifiziert den strukturellen Teil der Schema-Sprache, mögliche Objekttypen und ihre Beziehungen.

2. *XML Schema Part 2: Datatypes* spezifiziert die möglichen Inhaltstypen von Objekttypen, d.h. Datentypen für Elemente und Attribute sowie einen Mechanismus zur Definition benutzerspezifischer Datentypen.

Vorzüge von Schemata gegenüber DTDs

Ein Vorteil ist implizit schon erwähnt worden. Da Schemata dieselbe Syntax wie XML-Dokumente verwenden, müssen Schemata-Entwickler im Unterschied zu DTD-Entwicklern keine andere Syntax lernen. Allerdings ist das Beherrschen einer Schema-Sprache notwendig, womit dieses Argument an Überzeugungskraft verliert.

[2] W3C Candidate Recommendation: http://www.w3.org/TR/xmlschema-1/

Überzeugender ist das Argument, dass Parser, die XML-Dokumente gegen Schemata statt gegen DTDs validieren, mit einem reduzierten Satz syntaktischer Regeln auskommen. Diesen Regeln müssen XML-Dokumente ohnehin folgen, um wohlgeformt zu sein. Das vereinfacht die Parser-Algoritmik und kommt der Performance der Verarbeitung von Dokumenten zugute. Außerdem fördert eine geringere Größe und Komplexität des Codes und damit ein geringerer Speicherverbrauch die Verwendung von Parsern in mobilen Endgeräten wie PDAs und mobilen Telefonen.

Mit Hilfe von Schema-Sprachen lassen sich für Elementtypen und Attribute Datentypen definieren, die wie das obige Beispiel andeutet, weit über die Möglichkeiten von Dokumenttyp-Definitionen hinausgehen. Dieser Mehrwert ist mit Typdefinitionen wie in Datenbank- und Programmiersprachen zu vergleichen. Bei Datentypen kann es sich beispielsweise um Fließkommazahlen, Währungsangaben, URIs oder Ober- und Untergrenzen von Intervallen handeln.

Ein weiterer Vorteil ist die Möglichkeit zur Vererbung von Eigenschaften deklarierter Objekte, insbesondere der Vererbung von Inhaltsmoden oder Attributen. [Mic99] XML erlaubt zwar den modularen Aufbau von DTDs mit Hilfe von Parameter-Entitäten, welche aber nicht das bieten, was Vererbung leisten kann.

Zusammenfassend kann festgehalten werden, dass sich mit Hilfe von Schema-Sprachen Konzepte realisieren lassen, die die Funktionalität oder Handhabbarkeit von XML signifikant verbessern. Deshalb lohnt sich der Einsatz von Schemata besonders dort, wo präzise und detaillierte Dokumentenbeschreibungen notwendig sind. Als Nebeneffekt des Verzichtes auf eine syntaktisch eigenständige Deklarationssprache lassen sich Schemata wie XML-Dokumente mit Stylesheets verknüpfen. Dadurch lassen sich die für Dokumentation vorgesehenen Elemente, deren Inhalte sich erklärend auf Deklarationen beziehen, in geeigneter Weise darstellen.

3.3 Techniken zur Verarbeitung von XML

Um auf im XML-Format gesicherten Daten zuzugreifen, ist eine spezielle Software nötig, die auch XML-Parser genannt wird. Zwei unterschiedliche Techniken kommen dabei zur Anwendung. Diese sind DOM und SAX und gewähren dem Programmierer eine vollkommen unterschiedliche Zugriffsart zu seinen Daten, die sich in den verschiedenen Herangehensweisen zur Lieferung der Informationen begründen. Diese im folgenden beschriebenen Modelle werden meist beide als API in den Parsern implementiert, welche in vielen Programmiersprachen frei zur Verfügung stehen.

3.3.1 Document Object Model

Das Document Object Model [DOM] ist eine Anwendungsprogrammierschnittstelle (API) für gültige HTML- und wohlgeformte XML-Dokumente. Es definiert die logische Struktur von Dokumenten und stellt Funktionen bereit, um diese zu manipulieren. Mit DOM können Programmierer Dokumente erzeugen, in ihrer Struktur navigieren, Inhalt und Elemente hinzufügen, modifizieren oder löschen. DOM repräsentiert ein Objektmodell, das einem Baum von Knoten entspricht und auf der Struktur des XML-Dokumentes basiert. Der Programmierer erhält Verweise auf das Objektmodell, um es zu manipulieren.

Das W3C arbeitet an Level 3 der DOM-Spezifikation. Derzeit ist in vielen Parsern nur Level 1 implementiert, welches zur Erzeugung und Änderung von XML-Dokumenten voll einsatzfähig ist.

Wenn Dokumente hierarchisch strukturiert sind und günstig in einem Baum repräsentiert werden können, dann ist das Document Object Model eine geeignete Lösung für die Realisierung des Zugriffs auf diese Dokumente. Dieses Modell hält das erzeugte Objekt im Speicher, wodurch es für sehr große Dokumente nicht praktikabel ist.

3.3.2 Simple API for XML

Die von David Megginson entwickelte Simple API for XML [SAX] bietet den Zugriff auf die Informationen den XML Dokumentes als eine Sequenz von Ereignissen und wählt kein Standard-Objektmodell. Dadurch arbeitet SAX schneller, erfordert aber folgende Eigenleistungen des Programmierers:

1. Erstellung eines eigenen Objektmodells.
2. Erstellung einer Klasse von Funktionen (Dokumenthandler), die auf SAX Ereignisse hört, diese sinnvoll verarbeitet und das Objektmodell erzeugt.

Abhängig von den geparsten Tags des XML-Dokuments werden Ereignisse ausgelöst. SAX liefert beispielsweise ein Event für jedes öffnende Tag, jedes schliessende Tag, #PCDATA- und CDATA-Abschnitte und einige weitere. Diese werden vom Dokumenthandler in korrekter Reihenfolge des Auftretens interpretiert, um die darauf erforderlichen Funktionen auszuführen.

Die Anwendung von SAX ist in den Fällen angebracht, in denen der Programmierer ein eigenes Objektmodell erstellt. Falls das Mapping durch strukturierte Informationen leicht zu realisieren ist, kann die Simple API for XML eine Alternative zu DOM sein.

3.4 Visualisierung durch Stylesheets

Für Publikationszwecke werden semantisch strukturierte XML-Dokumente in Darstellungen konvertiert, die Online oder durch Druck veröffentlicht werden. Zum Formatieren von Daten wurden mehrere Stylesheet-Sprachen entwickelt. Um inhaltliche Beschreibungen in XML-Form zu visualisieren, werden sie mit Layout-Definitionen in Form von Stylesheets kombiniert. Die bekanntesten Stylesheet-Sprachen sind FOSIs, DSSSL, CSS und XSL und werden in den folgenden Abschnitten kurz vorgestellt.

FOSIs

Die Formatting Output Specification Instances (FOSIs) wurden vom U.S. Department of Defense entwickelt. Es war der erste Versuch, Stylesheets im Militärbereich zu standardisieren. Die Spezifikation ist in MIL-PRF-28001C (Markup Requirements and Generic Style Specification for Electronic Printed Output ans Exchange of Text) definiert. [Wal99]

DSSSL

Die Document Style Semantics and Specification Language [DSSSL] wurde von der ISO (International Organisation for Standardization) entwickelt. Als Ausgabeziele für die DSSSL-Transformationen sind sowohl Online- als auch Druckformate vorgesehen. DSSSL definiert eine Style-Sprache und eine Transformation-Language, die beide zu Formatierungsaufgaben verwendet werden. Die Sprache arbeitet über dem SGML-Format und schließt deshalb XML-Dokumente ein. Der komplette Prozess von den ursprünglichen SGML-Quellen bis hin zum fertig formatierten und darstellbaren Format verläuft über mehrere Schritte und ist relativ aufwendig.

CSS

Die W3C CSS Working Group erstellte CSS als eine Stylesprache für HTML. Cascading Style Sheets [CSS] werden in zwei Spezifikationen des W3C erläutert: CSS1 und CSS2. CSS1 wurde im Dezember 1996 veröffentlicht und erklärt ein einfaches Formatierungsmodell für eine Onlinepräsentation. Die Spezifikation CSS2 wurde im Mai 1998 fertiggestellt und baut auf CSS1 auf, indem alle Eigenschaften von CSS1 übernommen und zusätzliche hinzugefügt wurden.

3.4.1 Extensible Style Language

XML forderte eine neue Stylesheet-Sprache und erhielt sie in Form der Extensible Style Language [XSL], die weiterhin von der W3C XSL Working Group entwickelt wird. XSL ist eine Sprache zum Formulieren von Stylesheets, die angewendet auf ein XML-Dokument in der Regel ein Dokument modifizierter Struktur ergeben. Die Sprache besteht aus zwei Teilen, einer Sprache zur Transformation von XML-Dokumenten und einem Vokabular zur Spezifikation von Formatierungssemantiken. Beide Sprachen werden folgend näher dargestellt.

XSL Transformations

XSL Transformations [XSLT] ist eine Sprache zur Transformierung von XML Dokumenten in andere XML- oder Textdokumente und baut auf XSL auf. XSLT enthält Regeln zur Beschreibung der Transformation von einem XML Dokument in ein anderes Dokument. Diese werden in XSL bei der Definition von Stylesheets verwendet. Eine Nutzung von XSLT kann aber auch unabhängig von XSL erfolgen. Transformationen werden als wohlgeformte XML-Dokumente ausgedrückt. Die in XSLT beschriebene Transformation beschreibt Regeln zur Wandlung des Quellbaumes in einen Resultatbaum. Beim Erstellen des Resultatbaumes können Elemente vom Quellbaum gefiltert, umgeordnet und neue Strukturen hinzugefügt werden. XSLT gebraucht intern XPath [XPath] zur Selektierung von Elementen, zur bedingten Verarbeitung und zur Generierung von Text.

XSL Formatting Objects

XSL Formatting Objects (XSL-FO) ist die zweite Hälfte von XSL und beschreibt das exakte Aussehen von Dokumenten für den Leser. Im Allgemeinen verwendet ein Stylesheet die XSL-Transformations-Sprache, um ein semantisches XML-Dokument in ein neues XML-Dokument zu wandeln, das sich dem XSL-FO Vokabular bedient. XSL Formatting Objects bieten ein weiterentwickeltes Layout-Modell als HTML & CSS und sind für einen weiterreichenden Gebrauch entwickelt worden. Für die Zukunft ist geplant, dass Web-Browser semantische Daten in Verbindung mit XSL-FO direkt anzeigen sowie auch das druckbare PDF und andere Formate.

3.5 Zusammenfassung

Der Einsatz von XML gestattet eine Digitalisierung von Daten unterschiedlichster Bereiche in eine für jeden Computer lesbaren strukturierten Text, sodass der Computer verwendet werden kann, diesen zu speichern, zu bearbeiten, zu durchsuchen, zu übertragen, anzuzeigen und auszudrucken. Für die Verwendung von XML als Grundlage der internen Verarbeitung eines modernen Systems sprechen folgende Gründe:

1. XML-Dokumente basieren auf ASCII-Text und besitzen damit ein portables Datenformat.
2. Der Inhalt der Dokumente wird von der Darstellung getrennt.
3. Die Darstellung der Dokumente kann mittels Stylesheets unabhängig vom Datenbestand verändert werden.
4. Das flexible Datenformat und die semantische Auszeichnung erleichtern die Bearbeitung und Weiterverwendung der Dokumente.

Als Stylesheet-Sprache bietet sich die Extensible Style Language an, die durch eine Transformationssprache und einer Sprache zur Festlegung von Formatierungen alle Voraussetzungen für eine optimale Konvertierung von XML-Dokumenten in eine Vielzahl visualisierbarer Formen unterstützt.

Für Klassen von Dokumenten mit gleichen Inhaltstypen können XML-Anwendungen in Form von DTDs oder mächtigeren Schemata erstellt werden. Dabei lassen sich Verbindungen mit den in

Abschnitt 2.2.8 erstellten Dokumentenklassen finden. Der Einsatz von XML-Applikationen wird weiter verfolgt. Deshalb analysiert Kapitel 4 für einen Einsatz vorstellbare XML-Anwendungen.

Bei mehrfachem Zugriff erweist sich DOM als die effektivere Alternative gegenüber SAX, bei der das Dokument für jeden neuen Zugriff neu geparst werden müsste.

Kapitel 4

XML-Anwendungen

Auf der Suche nach geeigneten XML-Applikationen, die eine moderne Infrastruktur zur Erledigung von Geschäftsvorfällen unterstützen könnten, wird auf die Bereiche Publikation, Geschäftsaktivität, Formularanbindung und Digitale Signatur eingegangen. Die zur Dokumentenrepräsentation verwendeten XML-Anwendungen werden anhand der aufgestellten Kriterien des Abschnittes 4.1 betrachtet, um eine mögliche Struktur für die Geschäftsdokumente des URZ zu finden.

4.1 Verwendbarkeitskriterien

In diesem Kapitel werden aktuelle XML-Applikationen zur Unterstützung der Erzeugung von Geschäftsdokumenten im XML-Format untersucht. Auf Grund der Analyse der Geschäftsdokumente des URZ haben sich für jeden Dokumententyp strukturelle Eigenheiten ergeben. In der Gesamtheit lassen sich die Anforderungen der zu strukturierende Inhalte an bestehende XML-Anwendungen folgendermaßen angeben:

1. Anschriften
   - Absender
   - Ein oder viele (Serienbrief) Empfänger
   - Dritt- und Viertanschriften
2. Kontaktinformationen (Absender, Empfänger)
3. Artikelspezifikation (Nummer, Anzahl, Beschreibung)
4. Textinhalte
   - Betreff, Anrede
   - Text (inklusive Listen und Tabellen)
   - Gruß und Unterschrift
5. Individuelle Optionen (z.B. Inventarisierung, Kapitel)
6. Auswahlmöglichkeiten (z.B. Stellungnahmen)

Vorhandene XML-Applikationen werden auf diese Anforderungen hin überprüft. Allerdings spielen auch weitere Eigenschaften existierender XML-Anwendungen eine entscheidende Rolle, wie die nächsten Abschnitte verdeutlichen. Zunächst werden XML-Applikationen zur Erstellung und Weiterverarbeitung in druckfähige Dokumente betrachtet. Weiterhin wird auf Anwendungen im Business- und e-Commerce-Bereich eingegangen.

4.2 Veröffentlichung

Es existieren XML-Anwendungen, um zu veröffentlichende Dokumente geeignet in XML zu speichern und um die Verarbeitung in druckfertige Formate zu unterstützen. Ein heikles Thema für inhaltsbezogene Auszeichnungssprachen wie XML sind Gestaltungselemente wie Tabellen, weil in ihnen Gestaltungs- über Strukturaspekte dominieren und die Art des Inhaltes im Voraus nicht bestimmt werden kann. Die Bedeutung der Tabelleninhalte ist von ihrer Darstellung abhängig. Diese Darstellung verlangt normalerweise einen erheblichen Aufwand an nicht-inhaltsbezogenem Markup. In den meisten XML-Anwendungen werden Tabellen daher einerseits im Hinblick auf die Formatierung und andererseits im Hinblick auf ihre Erfassung betrachtet. Da die Darstellung von Tabellen in Geschäftsdokumenten des URZ einheitlich vorgegeben werden soll, sind Gestaltungselemente nicht notwendig.

4.2.1 DocBook

DocBook [Wal99] bietet ein System zum Schreiben von strukturierten Dokumenten unter Verwendung von SGML oder XML. DocBook ist eine in SGML spezifizierte Dokumententypdefinition, definiert eine Menge von Tags zur Beschreibung von Büchern, Artikeln und anderen Dokumentenarten und ist vor allem zur Erstellung technischer Dokumentationen geeignet. Die relativ weit verwendete DTD wird von einer Reihe kommerzieller und freier Werkzeuge unterstützt. DocBook ist auch als XML Version verfügbar, die eine eingeschränkte Version der mehr als 300 Elementen starken SGML DTD darstellt. Zur Erstellung von DocBook Dokumenten sei auf [Wal99] und die Verwendung von Katalogen, welche Standardmechanismen zur Auflösung von „Public identifiers“ in „System identifiers“ sind, verwiesen.

Verwendbarkeit der Spezifikation

DocBook Elemente können in folgende Kategorien eingeteilt werden:

- Sets, Books, Divisions, Components, Sections
- Meta-Informationen Elemente
- Block Elemente
- Inline Elemente

Eine Menge (die hierarchischen Wurzel von DocBook-Dokumenten) besteht aus Büchern, die ihrerseits aus Elementen wie Widmung, Navigationskomponenten, Divisions oder Komponenten aufgebaut sein können. Die hierarchische Strukturierung in Artikel und verschiedene Sektionen lassen einen ähnlichen Aufbau wie in LaTeX erkennen. Meta-Informationen enthalten inhaltsbeschreibende Informationen wie Autor und Titel. Block-Elemente beschreiben die Art von Absätzen oder Blöcken wie Listen oder Tabellen. Inline-Elemente werden zur Formatierung des laufenden Textes verwendet.

Mit DocBook lässt sich ein Dokument sehr gut im Sinne der Textformatierung strukturieren, was Funktionen wie das Anlegen von Inhaltsverzeichnissen, Indexen, Glossaren oder Literaturübersichten belegen. Das unterstreicht die hervorragende Nutzbarkeit für die vollständige Beschreibung von Büchern und Dokumentationen. Bei der Beschreibung der Geschäftsdokumente steht aber die semantische Struktur und nicht die Formatierung des Inhaltes im Vordergrund.

Druckgerechte Weiterverarbeitung

Um DocBook-Dokumente in einem druckbarem Format zur Verfügung zu stellen, kommen Stylesheets zum Einsatz. Für die Konvertierung gibt es verschiedene Werkzeuge. Das am meisten benutzte Tool ist Jade [Jade] von James Clark, was DSSSL Stylesheets verwendet. Es ist frei erhältlich und kann RTF, TeX, MIF und SGML erzeugen. Eine SGML zu SGML Transformation

ist zum Beispiel DocBook zu HTML. Eine Zusammenfassung weiter Tools ist im Anhang von [Wal99] enthalten.

Da die Tags für Geschäftsdokumente des URZ nicht ausreichen und in keinster Weise dafür ausgelegt sind, ist DocBook nicht für die Repräsentierung von Business-Dokumenten geeignet. Allerdings gibt es eine Möglichkeit, DocBook an eigene Bedürfnisse anzupassen. Man kann weitere Inline-Elemente und Attribute hinzufügen oder löschen und die Struktur vorhandener Elemente ändern. Das aufwendige Ändern der DTD wird Veränderungen der verwendeten Tools und Stylesheet nach sich ziehen und lohnt auf Grund der Komplexität von DocBook nicht.

4.2.2 Extensible Hypertext Markup Language

Die Extensible Hypertext Markup Language XHTML 1.0 [XHTML] ist eine Umformulierung von HTML 4.0 in eine XML 1.0 Anwendung. XHTML kann als Sprache für Inhalt verwendet werden, welche XML- und HTML 4-konform ist. Die Spezifikation definiert drei DTDs. Die Semantik der Elemente und ihre Attribute sind in der W3C Empfehlung für [HTML] bestimmt. Bei Einhaltung einiger weniger Regeln gewährleistet XHTML die Kompatibilität zu existierenden HTML Engines. XHTML-Dokumente sind XML-basiert und erstellt, um von XML-basierten Engines verarbeitet werden zu können. In XML ist es relativ einfach, neue Elemente und Attribute einzuführen. Die XHTML-Familie bietet Unterstützung für Erweiterungen der Sprache und kann diese in Form von Modulen anbinden. Sie ermöglichen eine Kombination von existierenden und neuen Features bei der Entwicklung von Inhalt und Engines. XHTML integriert HTML und verfügt damit über Tabellen und Listen, die in bestehende Strukturen von XML-Geschäftsapplikationen eingebunden werden könnten. XHTML ist keine Sprache zur Definition von Geschäftsdokumenten, sondern ein Standard zur Beschreibung von Webseiten. Die Darstellung ist auf das Web bezogen und hat keine Beziehung zu exakten druckfertigen Dokumenten.

4.2.3 Resource Description Framework

Das Modell des Resource Description Framework [RDF] ist ein einheitlicher Standard für Metadaten zur Beschreibung von Internet-Ressourcen und spezifiziert Möglichkeiten der Zusammenarbeit zwischen Anwendungen, welche maschinenverständliche Informationen im Web austauschen. RDF nutzt XML zum Austausch von Ressourcen-Beschreibungen beliebigen Typs im Web und definiert die Speicherung und Erfassung von Metadaten. Die RDF Schema Spezifikation [RDFSchema] spezifiziert ein Basistypensystem zur Verwendung von RDF Modellen und definiert Ressourcen und Eigenschaften. Sie bietet Informationen zur Interpretation von Eigenschaften und Aussagen, welche im RDF Daten Modell definiert wurden. RDF ist ein maschinenverständliches Metadatenformat, welches ein universelles Beschreibungsformat für Ressourcen im Netz darstellt. Zu den Einsatzmöglichkeiten gehören das Finden von Ressourcen im Web, Unterstützung von Suchmaschinen, Beschreibung von Inhalt und Inhaltsverbindungen verschiedener Websites oder digitalen Bibliotheken und Unterstützung von intelligenter Agentensoftware zur Bearbeitung von verteiltem Wissen. RDF ist auch zur Beschreibung von Rechten von Web-Seiten geeignet und bietet zusammen mit digitalen Signaturen eine Art „Web of Trust“. Das Resource Description Framework präsentiert ein Konzept, das ein System zur Erledigung von Geschäftsprozessen nicht tangiert.

4.2.4 Weitere XML-Anwendungen

Open eBook

Open eBook [OEB] 1.0 der „Open eBook Initiative“ bietet eine XML-basierte Spezifikation zur Repräsentierung des Inhaltes von elektronischen Büchern. Die Spezifikation definiert Regeln zur Sicherung von Genauigkeit, Zugriff und Präsentation von elektronischem Inhalt über verschiedene eBook-Plattformen. Die Spezifikation baut auf etablierten Standards zur Inhaltsformatierung auf. Sie kombiniert Teile von XML 1.0, XML Namensräumen, HTML 4.0, CSS1 und einigen weiteren

in der Art, dass die Erstellung, Organisation, Präsentation und der Austausch von elektronischen Dokumenten gewährleistet ist. Der Inhalt kann auf allen OEB-kompatiblen Lesesystem gesehen werden. Die OEB-Spezifikation definiert zwei DTDs, eine Package DTD und eine Basis OEB Dokumenten DTD. Das Package organisiert die komplette Publikation auf Lesesystemen und die zweite DTD implementiert einen Teil von HTML. Open eBook ist für Darstellungszwecke auf mobilen Endgeräten entworfen worden und hat damit keine direkte Verbindung mit druckfertigen Dokumenten. Die Spezifikation eignet sich nicht für die Definition von Geschäftsdokumenten.

CALS

CALS steht für „Continuous Acquisition and Lifecycle Support“ [Wal99] und ist ein Programm des US-Verteidigungsministeriums, das inzwischen globale Bedeutung über den militärischen Bereich hinaus gewonnen hat. Seit den 80er Jahren hat es die Integration von Geschäftsprozessen und die Anwendung von Standards bei der Entwicklung, Verwaltung und beim Austausch von Informationen zum Ziel. CALS-Tabellenmodelle werden vorherrschend bei technischen Dokumentationen verwendet. Es sind allerdings etliche Varianten des CALS-Modells im Umlauf. Neben den erwähnten XHTML- und CALS-Tabellenmodell existiert ein ISO-Tabellenmodell, das keine weitere Verbreitung gefunden hat.

Text Encoding Initiative

Die Text Encoding Initiative [TEI] beschäftigt sich hauptsächlich mit der Digitalisierung gedruckter Texte und dem Austausch dieser. Zu diesem Zweck wurde ein umfassendes XML-Dokumentenformat zur digitalen Erfassung von Dokumenten entwickelt. Im Vordergrund steht dabei die Erfassung und Archivierung von nichtdigitalen Formaten, und bietet eine Reihe von Elementen zur Formatierung des Inhaltes.

ISO 12083

ISO 12083 ist ein Standard, der von der International Standards Organisation (ISO), der National Information Standards Organisation (NISO) und dem American National Standards Institute (ANSI) veröffentlicht wurde. Es stellt einen Teil von SGML dar, welcher für elektronischen Markup von Büchern, Artikeln, Folgen und mathematischen Text benutzt werden kann. Für jeden angeführten Bereich gibt es eine 12083 Standard DTD. Eine DTD definiert eine Menge von Tags und Regeln zum Erstellen eines Dokumentes, sodass die Struktur und der Inhalt computerlesbar sind. ISO 12083 ist für Archivierung und Web-Publikation einsetzbar, für weitere Stadien der Veröffentlichung wie dem Druck jedoch kaum beachtet oder genutzt. Die von Tony Hicks vorgeschlagenen Verbesserungen [Hic98] der Spezifikation könnten zu einer breiteren Nutzung führen, ändern aber nichts an der Tatsache, dass sie auf Grund mangelnder Definitionsmöglichkeiten nicht für Geschäftsdokumente geeignet ist.

4.3 Geschäftsverkehr

Die Suche nach XML-Applikationen für deutsche Geschäftsdokumente, die nach den aufgeführten DIN-Normen denen des URZ entsprechen, bliebt erfolglos. Zur Zeit gelten die Bemühungen der Spezifikation von XML-Anwendungen für den Geschäftsverkehr fast ausschließlich dem Finanzsektor und e-Commerce.

4.3.1 Common Business Library

Commerce One's[1] Common Business Library [CBL] 2.0 war die erste offene XML Spezifikation (28. Juli 1999) für den Austausch von Businessdokumenten, welche sich den Vorteilen von

[1] Commerce One, Inc.: http://www.commerceone.com

Schemata bedient. Diese beinhaltet unter anderem Produktbeschreibungen, Kaufaufträge, Rechnungen und Lieferzeitpläne. Die Spezifikation ist als DTD, XDR und SOX verfügbar. Eine Bereitstellung im XML-Schema [XML-Schema] ist in Arbeit. CBL 2.0, auch als xCBL bekannt, spezifiziert eine Menge von XML-Baublöcken (Module) und ein Dokumentengerüst, das eine Erstellung verschiedenartiger e-Commerce-Dokumente erlaubt. CBL besteht aus Informationsmodellen für folgende Geschäftskonzepte:

1. Geschäftsbeschreibungen von Firmen, Diensten und Produkten
2. Geschäftsformulare wie Kataloge, Kaufaufträge und Rechnungen
3. Standardmaße, Daten und Zeiten, Standorte, Klassifikationscodes

Nachfolgendes Beispiel zeigt ein Geschäftsdokument, welches eine Rechnung repräsentiert. Das dabei verwendete Invoice-Modul deckt die strukturellen Ansprüche der Angebotseinholung bis auf die Texteingabe ab und kommt den oben definierten Erwartungen am nächsten. Weitere Module für e-Commerce Dokumente spezifizieren Kaufaufträge, Überprüfungen des Orderstatus, Preiskataloge und -checks sowie Verfügbarkeitschecks oder Produktkataloge.

```
<?xml version="1.0" ?>
<!DOCTYPE Invoice SYSTEM "CBL.dtd">
<Invoice>
  <InvoiceHeader>
    <InvoiceDate> 20000612 </InvoiceDate>
    <ContractNumber> xyz007 </ContractNumber>
    ...
  </InvoiceHeader>
  <InvoiceParties>
    <Buyer>
      <NameAddress>
        <Name1> Example.com </Name1>
        <Address1> Samplestreet </Address1>
        ...
      </NameAddress>
    </Buyer>
    <Supplier> ... </Supplier>
  </InvoiceParties>
  <ListOfInvoiceDetail>
    <InvoiceDetail>
      <BaseItemDetail>
        <LineItemNum> 1 </LineItemNum>
        <SupplierPartNum> ... </SupplierPartNum>
        <ItemDescription> Computer </ItemDescription>
        <Quantity> <Qty> 10 </Qty> ... </Quantity>
      </BaseItemDetail>
      <InvoiceUnitPrice> 800.00 </InvoiceUnitPrice>
    </InvoiceDetail>
    ...
  </ListOfInvoiceDetail>
  <InvoiceSummary> ... </InvoiceSummary>
</Invoice>
```

Invoice-Dokumente spezifizieren die am Geschäftsvorgang beteiligten Parteien, grundlegende Informationen über das Dokument und eine detailierte Aufführung über bestellte Waren sowie eine Zusammenfassung dieser.

4.3.2 Commerce XML

Commerce XML [cXML] ist seit dem 8. Februar 1999 ein XML Standard für den elekronischen Geschäftsverkehr im Business-zu-Business Bereich, der unter der Führung von Ariba[2] von mehr als 40 Firmen entwickelt wurde. cXML definiert Request/Response Prozesse für den Austausch

[2] Ariba, Inc.: http://www.ariba.com

von Transaktionsinformationen zwischen Käufer und Lieferer. Diese Geschäftsprozesse beinhalten Kauforders, Änderungsorders, Bestätigungen, Statusabfragen und -updates, Auslieferungsnotizen und Zahlungstransaktionen.

Im folgenden Beispiel werden Teile einer Kauforder vorgestellt, die von allen spezifizierten Geschäftsprozessen die größten Ähnlichkeiten mit den geforderten Voraussetzungen an XML-Anwendungen haben.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "cXML.dtd">
<cXML version="1.1" payloadID="id" timestamp="2000-06-12" ..>
  <Header> ... </Header>
  <Request deploymentMode="test">
    <OrderRequest>
      <OrderRequestHeader orderID="007" orderDate="2000-06-12" ..>
        <Total> <Money currency="USD"> 9120.00 </Money> </Total>
        <ShipTo>
          <Address>
             <Name xml:lang="en"> Example </Name>
             <PostalAddress name="foo">
               <DeliverTo> Mr. Sample </DeliverTo>
               <Street> Samplestreet </Street>
               ...
             </PostalAddress>
          </Address>
        </ShipTo>
         ...
      </OrderRequestHeader>
      <ItemOut quantity="10" requestedDeliveryDate="2000-06-12">
        <ItemID> <SupplierPartID> 12345 </SupplierPartID> </ItemID>
        <ItemDetail>
          <UnitPrice>
            <Money currency="USD"> 800.00 </Money>
          </UnitPrice>
          <Description xml:lang="en"> Computer </Description>
          ...
        </ItemDetail>
        <Comments xml:lang="en-US"> no Intel inside! </Comments>
      </ItemOut>
    </OrderRequest>
  </Request>
</cXML>
```

Die Spezifikation des Kaufauftrages ist allerdings wie auch bei CBL sehr umfangreich. Nach den Identitätsinformationen im *Header* werden im *OrderRequestHeader* die beteiligten Parteien, Versand-, Steuer- und Zahlungseigenschaften beschrieben. Der Inhalt des *ItemOut*-Tags behandelt einen Artikel in sehr detaillierter Form.

4.3.3 BizTalk Framework

Microsoft[3] veröffentlichte im September 1999 die Spezifikation des BizTalk Frameworkes 1.0 unter [BizTalk]. An der fortschreitenden Entwicklung waren Unternehmen und Organisationen als Zusammenschluß im BizTalk Steering Committee beteiligt. Das BizTalk Framework bietet eine Spezifikation zur Entwicklung von XML-basierten Kommunikationslösungen zwischen Anwendungen und Organisationen, die auf standardisierter Technologie wie XML, XML-Schema und MIME basiert.

Zum Austausch von Informationen wurden zwei Dokumententypen definiert. Das ist zum einen das *BizTalk Document* und zum anderen die darauf aufbauende *BizTalk Message*. Der zweite Dokumententyp wird zum Datentransfer zwischen BizTalk Servern verwendet. Die Spezifikation des *BizTalk Document* definiert eine Struktur wie in Abbildung 4.1 dargestellt. Während im *BizTalk Header* Informationen zu den beteiligten Geschäftspartnern untergebracht sind, beinhaltet der *Document Body* selbtändige Geschäftsdokumente. Das folgende Beispiel stellt einen Ausschnitt eines BizTalk Dokumentes dar, der dessen Verwendungszweck verdeutlicht.

[3] Microsoft Corp.: http://www.microsoft.com

```
<?xml version='1.0' ?>
<biztalk_1 xmlns="urn:biztalk-org:biztalk:biztalk_1">
  <header>
    <delivery>
      <message>
        <messageID>xzy007</messageID>
        <sent>2000-08-06T19:00:01+02:00</sent>
        <subject>Purchase Order</subject>
      </message>
      <to>
        <address>http://www.exapmle.com/recv.asp</address>
        <state> <referenceID/> <handle/> <process/> </state>
      </to>
      <from>
        <address>mailto:foo@supplier.com</address>
        <state>
          <referenceID>123</referenceID>
          <handle>7</handle>
          <process>myprocess</process>
        </state>
      </from>
    </delivery>
    <manifest>
      <document>
        <name>PO</name>
        <description>Purchase Order</description>
      </document>
    </manifest>
  </header>
  <body>
    <PO xmlns="x-schema:http://schema.sample.org/zi0124pf.xml">
      ...
    </PO>
  </body>
</biztalk_1>
```

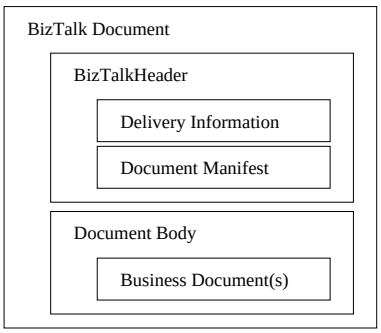


Abbildung 4.1: *BizTalk Document*

Das Wesen des *BizTalk Documents* liegt in der Definition eines Headers zur einfachen und schnellen Identifikation von Geschäftspartnern sowie der Art der integrierten Businessdokumente. Das BizTalk Framework versteht sich somit als ein Grundgerüst zur Bereitstellung von Routinginformationen, welche den Geschäftsverkehr über Unternehmensgrenzen hinaus optimieren soll.

4.3.4 Weitere XML-Businessanwendungen

Eine weitere interessante XML-Applikationen ist die von IBM veröffentlichte **Trading Partner Agreement Markup Language** (tpaML) [tpaML]. Die Grundlage von tpaML ist die Handelspartnervereinbarung (Trading Partner Agreement, TPA). Eine TPA ist ein elektronischer Vertrag, der XML zur Vereinbarung von allgemeinen Vertragsbedingungen und -konditionen, Mitgliedern (zum Beispiel Käufer und Verkäufer), Kommunikations- und Sicherheitsprotokollen und Geschäftsprozessen benutzt. Eine TPA definiert die Art des Transportes, des Dokumentenaustausches und der verwendeten Geschäftsprotokolle für Geschäftspartner. Auffallend ist die ausführliche Spezifikation von Mitgliederinformationen der TPA.

Weitere XML-Applikationen, wie Finance XML (finXML), Financial Products Markup Language (fpML), Open Financial Exchange (OFX), Electronic Commerce Modeling Language (ECML), Open Application Group (OAG) und Open Buying on the Internet (OBI) stammen eher aus dem Finanzbereich und konnten den gestellten Erwartungen nicht entsprechen.

4.4 Formulare

Webformulare sind ein wichtiger Teil des Web, erlauben Interaktion zwischen Lesern von Dokumenten und deren Autoren, Webseitenbesuchern und Webservern, Softwareprogrammen oder Käufern und Verkäufern über das Web.

4.4.1 XForms

Nach Meinung des W3C sind [XForms] die nächste Generation von Web-Formularen. Sie basieren auf XML-Schema [XML-Schema] und definieren Datentypen ähnlich Schema Part 2: Datatypes. Es werden reichere Web-Formulare benötigt, die eine größere Flexibilität und einen reicheren Interaktionsmechanismus erlauben. XForms sind das Resultat der Bemühungen um einen neuen Standard für Web-Formulare, eine neue plattform-unabhängige Markup-Sprache für die Benutzerinteraktion zwischen User-Agent und entfernter Schnittstelle.

XForms sind als Modul für [XHTML] 1.0 entworfen und bieten folgende Merkmale:

- Ein explizites Datenmodell, welches ein Formular als gemeinsamen Datentyp mit Richtlinien von und zwischen Formulardatenwerten spezifiziert
- Benutzerschnittstelle, eine Menge von Kontrollelementen
- Benutzung von XML und Unicode für den Austausch der Formulardaten mit Servern

Architektur

Es gibt eine feine Trennung zwischen dem Zweck des Formulars und der Präsentation des Formulars. Deshalb gibt es zwei Spezifikationen: „Data Model“ und „User Interface“. Das Datenmodell erlaubt die Definition einer abstrakten Struktur des Formulars ohne ein Benutzerinterface zu spezifizieren. Außerdem gibt es mit XForms eine neue Layer für die Benutzerschnittstelle mit größerer Interaktionsvielfalt. XForms erlaubt verschiedenen Benutzerschnittstellenmodulen die Interaktion mit dem selben Datenmodell.

Beispiel

Beim Entwurf der XForms wurde sowohl auf die Verbesserung des Web Formulars als auch auf die Datenweiterverarbeitung Wert gelegt. Einen Einblick in die Architektur von XForms soll folgendes Beispiel bieten. In diesem wird das Datenmodell zweckmäßig in XHTML eingebettet.

```
<?xml version="1.0"?>
<!DOCTYPE html SYSTEM "xhtml-xforms1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Address Example</title>
  <xform xmlns="http://www.w3.org/2000/xforms"
    action="http://www.sample.de/receive.cgi"
    method="postXML" id="test_form">
    <model>
      <group name="addressTest">
        <group name="receiver">
          <string name="name"/>
          <string name="street"/>
        </group>
      </group>
```

```
    </model>
    <instance>
      <addressTest>
        <receiver>
          <name>Mr. Smith</name>
          <street>Samplestreet</street>
        </receiver>
      </addressTest>
    </instance>
  </xform>
</head>
<body>
  <form name="test_form">
    Name   : <input name="addressTest.receiver.name"/><br/>
    Street : <input name="addressTest.receiver.street"/><br/>
    <button onclick="submit('test_form')">Submit</button>
  </form>
</body>
</html>
```

Das XForm Datenmodell ist durch das Tag `model` gekennzeichnet und kann durch ein XML Originaldokument repräsentiert werden, welches in gleicher Struktur auch zum Austausch mit dem Webserver bestimmt ist. Wenn der Nutzer die Einträge im Texteingabefeld ändert, werden die instanziierten Daten aktualisiert. Bei Betätigung des Submit-Knopfes werden die Daten zum Beispiel als XML Dokument zum Server gesendet. Auf der Serverseite kann das erhaltene XML Dokument mit dem Schema für das Originaldokument validiert werden.

4.4.2 XML Forms Architecture

Die XML Forms Architecture [XFA] von JetForm[4] bietet spezielle Formularfunktionalitäten für Nutzer und Anwendungen. XFA ist eine offene Spezifikation, welche definiert, wie ein Formular aussehen und als XML-Dokument verarbeitet werden soll, und zwar bei einer Separierung der Datenelemente von Details der grafischen Präsentation. XFA-Template ist eine XML-basierte Sprache der XFA zur Modellierung von elekronischen Formularen. Dabei werden Bedürfnisse wie Sicherheit, Präsentation, Austausch, Weiterverarbeitung, Ausgabe und Druck von Informationen unterstützt. Die Spezifikation beschreibt eine Konstruktion von sicheren Formularen mit hoher Detailgenauigkeit, automatisierter Validierung, einbindbaren Nutzerinterfacekomponenten und flexiblen Datenmanagement.

XFA unterscheidet klar zwischen zwei Sichten auf Formulare:

- Formular, mit dem der Formularnutzer arbeitet
- Template, welches der Formulardesigner erstellt

JetForm's XML Forms Architektur ist eine XML-Applikation für elektronische Formulare und beschreibt:

1. Unterschiedliche Sichten auf das Dokument und die Daten (Bildschirm, Druck, Web ..).
2. Unterstützung für Vereinigung von Formular und Daten, und der Trennung von Formular und Daten.
3. Absolute und relative Positionierung von Formularobjekten und -daten.
4. Unterstützung von mehreren Skript- und Kalkulationsengines.
5. Unterstützung von digitalen Signaturen.
6. Die Verwendung von XML Datendefinitionen zur Interoperabilität.

[4] JetForm: http://www.jetform.com

XFA besitzt eine komplett eigene Syntax und ist sehr genau bei der Definition von Positionen und Formatierung wie Ausrichtung und Größenangaben. Es werden sehr viele Datentypen angeboten, darunter auch grafische wie Linien und Rechtecke. XFA erlaubt den Zugriff auf Formulardaten per XFA Scripting Object Model (XFA-SOM), was logisch mit dem [DOM] vergleichbar ist. Weiterhin werden durch die Syntax unter anderem Validierungen, Berechnungen und Ereignisse unterstützt, wie folgende Ausschnitte zeigen:

```
...
<Field Name="Quantity">
  <Validate>
    <Script>Within($, 0, 19)</Script>
  </Validate>
</Field>
...
<Field Name="ShipDate">
  <Calculate>
    <Script>Num2Date(Date() + 2, DateFmt())</Script>
  </Calculate>
</Field>
...
<Field Name="Button1">
  <Caption>
    <Text>Clear</Text>
  </Caption>
  <Event HandlerFor="Click">
    <Script>
      <![CDATA[ Quantity[*] = Null() ]]>
    </Script>
  </Event>
</Field>
...
```

Wie die Beispiele schon zeigen, ist XFA-Template eine umfassende Sprache zur präzisen Definition von Formularen. Allerdings ist eine Implementation auf Basis dieser Spezifikation sehr umfangreich.

4.4.3 Extensible Forms Description Language

Die Extensible Form Description Language [XFDL] ist eine Sprache zur Erstellung von legalen und verbindlichen XML Dokumenten. XFDL bietet Komponenten für eCommerce Lösungen, wie die Fähigkeit zum sicheren Versand und Empfang von XML-Dokumenten. Als eCommerce Sprache hat sie weiterhin die Fähigkeit zur Kapselung von Präsentation, Daten, konzeptueller Logik und Geschäftssemantik in einem einzelnen Dokument. Die Dokumente können digital unterschrieben und gesichert werden.

Die Spezifikation XFDL 4.0 liegt seit dem 2. September 1998 als W3C NOTE vor und ist zur Diskusion freigegeben. Sie wurde von UWI[5] und Tim Bray entwickelt. Der Zweck von XFDL ist die digitale Repräsentation komplexer Formulare, wie sie im Geschäftsleben oder Regierungskreisen verwendet werden. Die Forderungen beinhalten ein hoch präzises Layout, Unterstützung von Dokumentation, integrierte Berechnungen und Eingabevalidierung, mehrfache überlagernde digitale Signaturen und legale, verbindliche und überprüfbare Transaktionsrecords. Das gesamte Formular wird als eine einzelne Einheit betrachtet. Das Formular-Template, die Daten und die interne Logik werden in einer einzigen Datei gesichert, die digital signiert werden kann. XFDL bietet unter anderem Built-In Logik, Kalkulationen, Typprüfungen, Anlagen und Online-Hilfe. Während das W3C die Trennung von Inhalt und Präsentation befürwortet, kombiniert XFDL Inhalt, Layout, Aktionen und digitale Signaturen.

XFDL ist eine XML-Implementation und spezifiziert ein offenes Protokoll, welches komplexe Geschäftsformulare in XML Syntax präsentiert. Die aktuellen Entwicklungsziele von XFDL sind der Entwurf einer Sprache, die

[5] UWI Unisoft Wares Inc.: http://www.UWI.com

1. Formulare als einzelne Objekte ohne Abhängigkeiten zu externen Definitionen präsentieren,
2. aus menschenlesbarem Text besteht,
3. ein offener Standard ist,
4. eine Syntax für interne mathematische und konditionale Ausdrücke bietet,
5. einen Anhang von willkürlicher Größe und Base-64 kodierter Binärdateien erlaubt,
6. präzises Layout zur Präsentation und zum Druck der Formulare anbietet,
7. die Server-seitige Verarbeitung durch Client-seitige Eingabevalidierung und -formatierung erleichtert,
8. Erweiterungen von eigenen Gegenständen, Optionen und Funktionen erlaubt,
9. umfassende Unterstützung für digitale Signaturen bietet, inklusive:
   - Abdecken des gesamten Inhaltes von geschäftlichen Transaktionen.
   - Mehrere Unterzeichner.
   - Verschiedene Unterzeichner (möglicherweise überlagernd) für Teile des Formulares.
   - Einfrieren der unterzeichneten Teile.

XFDL spezifiziert im allgemeinen einen ähnlichen Funktionsumfang zur Definition von Formularen wie XFA, bietet keine Positionierung aber eine Vielzahl von neuen Formularauszeichnungselementen. Es lassen sich beispielsweise dokumentbezogene Verarbeitungsregeln in das Dokumenten einbetten oder das Ausführen von Aktionen bestimmen wie unterhalb dargestellt. Die Aktion soll automatisch aller 10 Minuten (600 Sekunden) eine Statusmeldung zum Server senden.

```
<action sid="sendStatus_action">
  <delay content="array">
    <ae>repeat</ae>
    <ae>600</ae>
  </delay>
  <type>submit</type>
  <url content="array">
    <ae>http://www.sampleserver.com/cgi-bin/recv_status</ae>
  </url>
</action>
```

4.5 Die Digitale Signatur

Die Anerkennung digitaler Signaturen ist die Grundlage effizienter elektronischer Geschäftsabwicklungen. Nach einer kurzen Einführung zur digitalen Signatur wird auf die vom W3C ausgearbeiteten XML-Anwendung eingegangen.

Digitale Signaturen sichern die Integrität und Authentizität unterschriebener Dokumente und gestatten einen Urhebernachweis. Integrität ist der Nachweis für die Unversehrtheit der Daten und Authentizität bestätigt die Echtheit und Glaubwürdigkeit der Daten.

So wie die Unterschrift per Hand für den Ursprung eines Briefes bürgt, bestätigt das elektronische Siegel den Verfasser oder den Unterzeichner eines elektronischen Dokumentes. Durch digitale Signaturen kann im elektronischen Geschäftsverkehr Rechtssicherheit [Gei98] erreicht werden.

Digitale Signaturen werden wie in Abbildung 4.2 dargestellt, mit dem geheimen/privaten Schlüssel generiert und mit dem öffentlichen Schlüssel verifiziert. Sie erfordern ein Public-Key-Kryptosystem, bei dem öffentliche und private Schlüssel existieren. Öffentliche Schlüssel müssen authentisch sein, was durch Zertifikate sichergestellt wird. Digitale Signaturen sind vom Inhalt

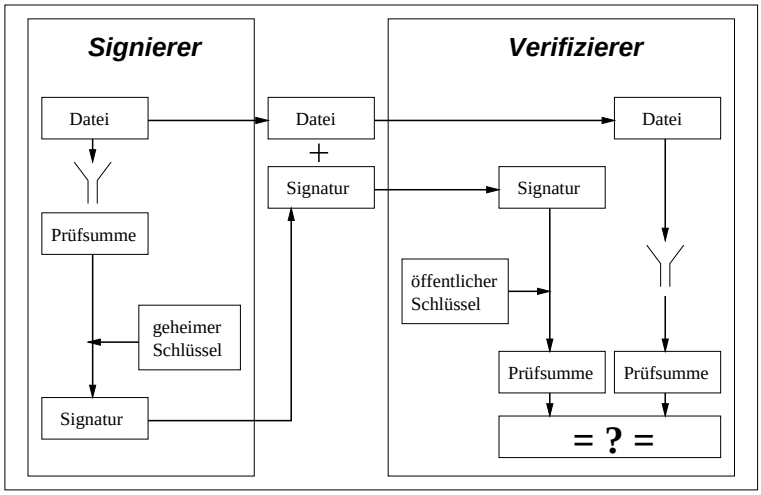


Abbildung 4.2: Die digitale Signatur

des unterschriebenen Dokumentes abhängig. Das Geschäftsdokument wird normalerweise nicht selbst, sondern der Hashwert (Prüfsumme oder Fingerabdruck) davon signiert. Ein Hashwert ist eine kurze eindeutige zu einer Nachricht passende Zeichenkette, der durch eine Hashfunktion erzeugt wird. Da der Hashwert sehr kurz ist, ist die Erstellung der Signatur über den Hash effizienter als das Unterschreiben des gesamten Dokumentes.

Ablauf

Ein Programm des Signierers verwandelt die Prüfsumme des Dokumentes mit einem geheimen Schlüssel in eine per Zeichenkette dargestellte digitale Signatur, die dann zur Botschaft gehört und mit ihr verschickt werden könnte. Aus der Signatur errechnet die Software des Empfängers mit dem öffentlichen Schlüssel des Verfassers die Prüfsumme und vergleicht sie mit der Prüfsumme der ursprünglichen Nachricht. Wenn beide identisch sind, stammt das Dokument vom Inhaber des öffentlichen Schlüssels.

4.5.1 Rechtliche Grundlage

Am 30. November 1999 wurde eine neue Regelung zur Abwicklung elektronischer Geschäfte von der Europäischen Union ausgearbeitet und im EU-Amtsblatt L13/12 vom 19. Januar 2000 veröffentlicht und damit in Kraft gesetzt. [EUSig] Diese Richtlinie müssen die Mitgliedsstaaten der EU innerhalb von 18 Monaten in nationales Recht umsetzen. Der von der EU beschlossene Standard legt die Anforderungen an die Erstellung von Signaturen sowie deren Haftungsansatz fest und soll dazu dienen, die Interoperabilität von Produkten für elektronische Signaturen zu fördern und der digitalen Signatur europaweit die gleiche Rechtsgültigkeit wie der handschriftlichen Unterschrift zu geben.

Die Richtlinie definiert verschiedene Begriffe im Zusammenhang mit den Sicherheitsabstufungen. [Fra99] Die *elektronische Signatur* bezeichnet Daten, die anderen elektronischen Daten beigefügt oder logisch mit ihnen verknüpft sind und die zur Authentifizierung dienen. Die *fortgeschrittene elektronische Signatur* ist ausschließlich einem Unterzeichner zugeordnet und ist so mit den Daten verknüpft, dass eine nachträgliche Veränderung erkannt werden kann. Schließlich geht es noch um *qualifizierte Zertifikate*, also Bescheinigungen mit den Signaturprüfdaten einer Person (Codes oder öffentliche kryptografische Schlüssel), die von einem sogenannten Zertifizierungsdiensteanbieter bereitgestellt werden.

Deutschland besitzt seit dem 13. Juni 1997 eine Signaturgesetz [SigG], das den von der EU beschlossenem Standard schon weitestgehend unterstützt. In diesem werden Rahmenbedingungen für digitale Signaturen geschaffen, unter denen diese als sicher gelten und Fälschungen digitaler Signaturen oder Verfälschungen von signierten Daten zuverlässig festgestellt werden können.

4.5.2 XML-Signature Syntax

Die W3C XML Signature Working Group arbeitet zusammen mit der IETF an einer XML-Anwendung für digitale Signaturen mit dem Namen *XML-Signature Syntax and Processing* [DSig]. Die in diesem Abschnitt vorgestellte Spezifikation ist der Working Draft vom 1.Juni 2000. Er spezifiziert die XML Syntax und Verarbeitungsregeln zur Erstellung von digitalen Signaturen. Sie liegt als XML-Schema und auch als DTD vor.

Digitale Unterschriften werden in XML durch das Element `Signature` repräsentiert und haben folgende Struktur:

```
<Signature>
  <SignedInfo>
    (CanonicalizationMethod)?
    (SignatureMethod)
    <Reference (URI=)? >
      (Transforms)?
      (DigestMethod)
      (DigestValue)
    (</Reference>)+
  </SignedInfo>
  (SignatureValue)
  (KeyInfo)?
  (Object)*
</Signature>
```

(`?` bedeutet ein oder kein Vorkommen; `+` bedeutet ein oder mehr Vorkommen; `*` bedeutet kein oder mehr Vorkommen)

XML Signaturen können auf jeden digitalen Inhalt (Datenobjekt) angewandt werden. Es besteht die Möglichkeit, eine XML Signatur für den Inhalt von einer oder mehreren Ressourcen zu erstellen. Signaturen beziehen sich auf Datenobjekte per URI. In einem XML Dokument sind Singnaturen zu lokalen Datenobjekten per Identifikator zugehörig. Solche lokalen Daten können in einhüllenden Signaturen integriert sein oder Signaturen selbst einbetten. Somit sind solche Signaturen über die Daten im selben XML Dokument erzeugt, welches auch die Signatur enthält. Einzeln stehende Signaturen unterschreiben externe Daten (Netzwerkressourcen).

Ein ausfürlich beschriebenes Beispiel und weitergehende Informationen über Syntax, Verarbeitungsregeln und zu verwendente und empfohlene Algorithmen sind in der W3C Spezifikation [DSig] angegeben.

Canonical XML

Unter XML in kanonischer Form versteht man eine physikalische Repräsentation eines XML Dokumentes. Die kanonische Form ändert sich auch unter syntaktischer Abweichung des XML Dokumentes nicht; vorausgesetzt die Abweichungen sind logisch äquivalent nach [XML] 1.0 Recommendation. Die Spezifikation „Canonical XML“ [CanXML] beschreibt eine Methode zur Erzeugung einer solchen Form. Wohlgeformte XML Dokumente werden unter Verwendung von [XPath] Ausdrücken in eine kanonische Form umgewandelt. „Canonical XML“ ist zur Verwendung durch Applikationen gedacht, die testen, ob sich ein Dokument nicht im Sinne der logischen Äquivalenz geändert hat.

4.6 Zusammenfassung

Die untersuchten XML-Anwendungen in den Bereichen Publikation/Digitalisierung und Geschäftsprozesse konnten den Erwartungen an eine geeignete Struktur der Geschäftsdokumente nicht gerecht werden, brachten aber interessante Impulse zur Entwicklung eigener Strukturen. Ein Lösungsansatz könnte in der Verwendung von Teilen der analysierten XML-Applikationen liegen, wie z.B. XHTML für die Textbereiche, cXML für Adressen und DSig für digitale Signaturen. Da dies aber nur Bruckstücke der erforderlichen Strukturen sind und weitere Teile hinzugefügt werden müssen, ist die Idee wenig praktikabel. Ferner wäre eine Erweiterung des zu erstellenden Systems um neue Geschäftsdokumente und eine einfache Implementation und Anwendung erschwert.

Die XML-Applikationen des Abschnittes 4.2 definieren fast ausschließlich die Darstellung zu digitalisierender Dokumente. Diese wurden betrachtet, da es zum einen in Geschäftsdokumenten Textbereiche gibt, in denen auch Darstellungselemente notwendig sind und zum anderen sollen die Dokumente einfach in ein druckbares Format übertragen werden, wobei XML-Applikationen zur Publizierung helfen könnten.

Alle Business-Anwendungen des Abschnittes 4.3 zeigten teilweise Übereinstimmungen mit den geforderten Kriterien (Absch. 4.1) und sind sehr umfassend, repräsentieren aber keine Geschäftsdokumente. In naher Zukunft sind Spezifikationen für Geschäftsdokumente im deutschen Raum mit hoher Wahrscheinlichkeit zu erwarten, da die Wirtschaft das Potential von XML erkannt hat. Erste deutsche Entsprechungen nach amerikanischen Vorbildern finden sich wiederum im e-Commerce-Bereich, z.B. einer Katalogspezifikation BMEcat. Auch das DIN wurde im Bereich XML aktiv und veröffentlichte im September 1999 (Neufassung im Juni 2000) den Norm-Entwurf DIN 16557-4 „XML/DTDs“. Damit hatte eine offizielle und neutrale Normungsorganisation erstmals einen normativen Vorstoß zum Thema XML gewagt, der als „Preliminary Work Item“ bei der ISO registriert wurde. Die E DIN 16557-4 legt aber lediglich einige Mapping-Regeln zur Überführung von EDIFACT-Dateien in XML-Strukturen und umgekehrt fest.

Die Formular-Anwendungen wurden untersucht, um Ansätze für die elegante Umsetzung von Daten des Web-Formulars in XML zu gewinnen. Sie beschreiben das Erscheinungsbild des Formulars und die Struktur der daraus resultierenden XML-Dokumente. Es zeichnet sich ab, dass XForms in einiger Zeit die herkömmlichen HTML-Formulare ablösen. XForms werden von einer Arbeitsgruppe des W3C entwickelt und lassen derzeit noch Fragen offen, z.B. File-Upload und die Digitale Signatur. Die XFA und die XDFL haben einen erheblich größeren Funktionsumfang als die XForms. XFA ist ein Template, das Felder definiert, deren Namen und Verwendungszweck vom Programmierer bestimmt werden. XDFL besitzt eine große Anzahl von Elementen zur Auszeichnung von XML Dokumenten. Die beiden letzteren XML-Anwendungen sind sehr komplex und erfordern zur Verarbeitung ein dementsprechendes System.

Ein modernes Verarbeitungssystem für Geschäftsprozesse sollte auch digitale Signaturen unterstützen. In Abschnitt 4.5.2 wird die XML-Signatur-Spezifikation des W3C angesprochen, die vor allem auf Online-Dokumente angewendet werden soll und auch sonst für den Gebrauch im URZ sehr überdimensioniert erscheint. Sie bietet einen Ansatz für weitere Überlegungen des Einsatzes einer Digitalen Signatur im elekronischen Büro.

Auf Grund der rasanten Entwicklung in diesem Bereich ist es durchaus möglich, dass mittlerweile in diesem Kapitel betrachtete oder neue XML-Applikationen zur Anwendung kommen. In der Arbeit konnten die aktuellen Entwicklungen nicht mehr berücksichtigt werden. Letztendlich werden die untersuchten XML-Anwendungen als Anregungen in die weitere Arbeit einfließen aber nicht als Basis für die interne Verarbeitung verwendet.

Kapitel 5

Entwurf des Systems

In diesem Kapitel werden Schwerpunkte des Entwurfs des Gesamtsystems wie Architektur, Datenstruktur und Nutzerschnittstelle dargelegt. Die Aufgabenstellung der Diplomarbeit enthält grundsätzliche Anforderungen an das zu erstellende System. Einerseits sollte die interne Verarbeitung der Anwendung auf XML basieren und andererseits ist eine Web-basierte Nutzerschnittstelle angedacht.

Der Einsatz der Extensible Markup Language bestimmt richtungweisend die Art der Verarbeitung von Daten und die Art der Datenstruktur von Dokumenten. Die im Abschnitt 3.1.3 aufgeführten Eigenschaften und Vorteilen belegen die sehr gute Eignung von XML zur Repräsentation und internen Verarbeitung von strukturieren Dokumenten. Ein Beispiel für das Verwenden von XML als Dateiformat ist das Textsatzsystem StarOffice, für das eine Projektgruppe[1] von Sun Microsystems existierende Dateistrukturen und -formatierungen durch das Definieren von XML-Tags in XML-Anwendungen umsetzt und das Ziel verfolgt, in naher Zukunft auf das alte Binärformat verzichten zu können.

Durch den Gebrauch einer Web-basierten Nutzerschnittstelle wird die Dateneingabe per Browser über Web-Formulare erfolgen. Eine Verwendung dieser Formulare erfolgt unabhängig vom Browser und Betriebssystem, was gegenüber systemspezifischen Produkten einen bedeutenden Vorteil darstellt. Die Installation der verwendeten Applikationen ist nicht nötig. Diese werden per Nutzerschnittstelle online angesprochen und liefern das gewünschte Ergebnis. Großer Beliebtheit erfreuen sich beispielsweise Portale zur Verwaltung persönlicher Emails, die auf Grund der erwähnten Vorteile so stark genutzt werden.

Das System wird unter Beachtung der systemtechnischen und technologischen Gegebenheiten der vorhandenen Infrastruktur des URZ konzipiert. Als systemtechnische Voraussetzung kann im Campusnetz der TU auf Linux-basierte Personal-Computer zurück gegriffen werden, die sich hervorragend zur Schaffung moderner Applikationen eignen. Somit basiert die Entwicklung des Prototyps auf dem Betriebssystem Linux, welcher dadurch leicht in die bestehende Infrastruktur des Rechenzentrums integriert werden kann. Der Schwerpunkt der Arbeit liegt in der Entwicklung einer attraktiven und sicheren Anwendung zur Erledigung von Geschäftsvorfällen mittels elektronischer Formulare. Dieses System zur Rationalisierung der Büroarbeit erhält den Namen „Elektronisches Büro“; in moderner gebräuchlicher Internet-Kurzform lautet dieser „eOffice“.

5.1 Entwurf der Architektur und Datenstruktur

Das zu entwickelnde System stellt eine Web-Erweiterung dar, da es Daten verarbeitet, die über eine Web-Schnittstelle eingegeben wurden. Es wird die Funktionalität des Intranets des Rechenzentrums um eine Anwendung zur Erledigung von Geschäftsvorfällen erweitern. In Abschnitt 5.3 werden die entworfenen Nutzerschnittstellen beschrieben, die zur Eingabe von Daten auf Web-Formularen basieren. Im Weiteren werden Varianten der Architektur des Gesamtsystems erörtert,

[1] OpenOffice.org: http://xml.openoffice.org/

die den technologischen Gegebenheiten des URZ genügen. Abschnitt 5.1.2 beschäftigt sich näher mit dem Entwurf des elektronischen Büros in Bezug auf Architektur und Speicherkonzept.

5.1.1 Auswahl der technologischen Grundlage

Web-Erweiterungen können in zwei Gruppen klassifiziert werden. [Hue98] Das sind zum einen die Erweiterungen auf der Klientenseite und zum anderen die auf der Serverseite, die beide mit ausgewählten Varianten folgend aufgelistet sind.

1. Erweiterungen auf der Klientenseite
   - Interpreter für Skriptsprachen
   - Laufzeitumgebung für übersetzte Programme
   - externe Viewer
   - Plugins
2. Erweiterungen auf der Serverseite
   - Common Gateway Interface
   - herstellerspezifische Modulschnittstellen
   - Servermodule zur Interpretation von Dokumenteninhalten

Das Ausführen von Applikationen der Klientenseite impliziert, dass die Rechenarbeit zur Verarbeitung von Eingabedaten auf den Rechnern der Nutzer geschieht. Auf Grund dieses primär negativen Aspektes sind solche Erweiterungen als Systemgrundlage nicht geeignet. Diese Anwendungen, wie beispielsweise Java-Applets, beanspruchen lange Ladezeiten, um ihre Arbeitsumgebung auf den Klientenrechnern einzurichten. Für das zu erstellende System werden Pakete zur Verarbeitung von XML nötig sein, die sehr umfangreich sind und die Ladezeit weiter erhöhen. Derartige Wartezeiten sind für den Nutzer und für ein attraktives System nicht akzeptabel. Eine unzureichende Performance eines Systems ist besser durch den Austausch eines Servers zu beheben, als durch das Ersetzen der Klientenrechner. Für das zu erstellende System ist weiterhin der Zugriff auf Daten eines Server unabdinglich, der sowieso durch eine Server-Applikation realisiert werden müsste.

Externe Viewer oder Plugin-Erweiterungen erfordern vom Nutzer einen Mehraufwand für Installationen und bestätigen die eben aufgeführten Nachteile von Erweiterungen auf der Klientenseite. Die Verwendung eines Interpreters für Skriptsprachen ist auf der Klientenseite allerdings empfehlenswert und beansprucht nur geringe Ressourcen des Nutzerrechners. Das übliche JavaScript kann zum Beispiel sehr gut zur Kontrolle der Formulareingaben verwendet werden.

Erweiterungen auf der Serverseite haben den Vorteil, dass die Verarbeitung der Daten vollständig auf einem Server geschieht und die Klienten nicht belasten. Sehr häufig werden CGI-Programme zur Verarbeitung von Nutzerdaten verwendet.

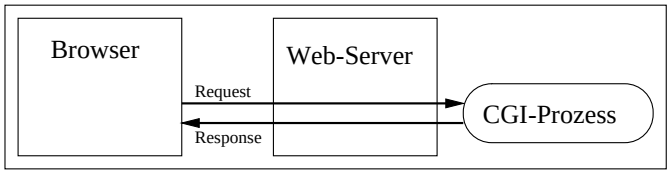


Abbildung 5.1: CGI-Programmierung

Das **Common Gateway Interface** [CGI] ist ein Protokoll, das einem Web-Server erlaubt, Daten eines Dokumenten-Requests von einem Browser an ein externes Programm durchzureichen

und das Ergebnis des Programmes als Antwort auf die Anfrage an einen Klienten zurückzusenden. (Abbildung 5.1) Typischerweise wird für jede Anfrage ein neuer Systemprozess gestartet. Unter Verwendung von CGI-Programmen werden Web-Dokumente dynamisch zur Anfragezeit erzeugt. Sie können in verschiedenen Sprachen implementiert werden, z.B. stehen dafür C/C++, Perl, Tcl oder Shells zur Auswahl.

Bei einer Dokumentenanfrage werden dem Programm Informationen über den Nutzer und Daten über die Anfrage selbst mittels Umgebungsvariablen zur Verfügung gestellt. Als Anfragemethoden kommen meist `HTTP-GET` oder `HTTP-POST` zum Einsatz. Unter Verwendung von `GET` werden die übergebenen Daten in der Umgebungsvariablen `QUERY_STRING` abgelegt, `CONTENT_TYPE` auf den MIME-Typ der Daten und `CONTENT_LENGTH` auf die Länge der Eingabedaten in Bytes gesetzt. Durch die Nutzung einer `POST`-Anfrage leitet der Server die Daten direkt an die Standardeingabe des CGI-Programms weiter. Wenn sich der Nutzer erfolgreich an einem Web-Server authentifiziert hat, wird der Nutzername in der Umgebungsvariablen `REMOTE_USER` bereitgestellt. Bei Apache-Servern kann eine Zugriffsbeschränkung auf Verzeichnisse durch das Anlegen von .htaccess-Dateien erreicht werden. Die Rückgabe eines erstellten Dokumentes erfolgt durch das Schreiben der Daten an die Standardausgabe des CGI-Programmes nach dem Setzen des Inhaltstypes.

Von den aufgeführten Sprachen zur Implementierung von CGI's bietet sich vor allem Tcl auf Grund der sehr einfachen und gut handhabbaren Syntax an. Tcl gewinnt im Zusammenhang mit dem Web immer mehr an Bedeutung und kann entweder als alleinstehende Anwendung oder eingebettet in Applikationprogrammen verwendet werden. Die Performance-Unterschiede und damit die Zusatzrechenzeit durch die Skriptausführung gegenüber übersetzten Programmen fallen auf Grund schneller Rechner nicht mehr ins Gewicht. Bei Übertragungen von Daten über das Internet sind sie wegen der dominierenden Übertragungszeiten irrelevant. [Bal99]

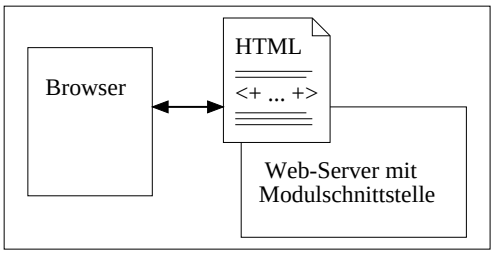


Abbildung 5.2: Microscripting

Gegenüber CGI's haben herstellerspezifische Modulschnittstellen und Module für serverseitige Skriptsprachen einen beachtlichen Performancevorteil, da diese Techniken nicht für jede Anfrage einen neuen Prozess erzeugen. (Abbildung 5.2) Von Netscape stammt beispielsweise die API-Schnittstelle, von Microsoft die ISAPI-Schnittstelle. Beide Schnittstellen sind für die Server-Software-Produkte der jeweiligen Hersteller optimiert. Ein entscheidender Vorteil der CGI-Schnittstelle bleibt jedoch die Tatsache, dass es sich um einen kommerziell unabhängigen, kostenlosen und produktübergreifenden Standard handelt. Serverseitige Skriptsprachen werden schon seit einigen Jahren in verstärktem Maße eingesetzt. Sie zeichnen sich durch das Einfügen von Skriptbefehlen in HTML-Seiten aus. Die eingebetteten Befehle werden abgearbeitet und die so dynamisch erzeugte Web-Seite an den Browser gesandt. Stellvertretend ist in diesem Zusammenhang der Hypertext-Prozessor PHP[2] zu sehen. Auch für länger bestehende Programmiersprachen existieren mittlerweile Erweiterungen auf diesem Gebiet. Der Programmierstil wird auch als Microscripting bezeichnet und entspricht wiederum dem Einfügen von Programmanweisungen zwischen HTML-Code. Für Tcl gibt es beispielsweise das Apache-Modul `mod_dtcl` [ModTcl] und sogenannte Microscripting-Web-Server wie den NeoWebServer[3] oder den eigenständigen Tcl Web Server [Bal99], die ein Microscripting von HTML-Dokumenten erlauben.

Auch für Java existiert eine Web-Scripting-Technologie. [Kus99] Die Java Server Pages (JSP) ähneln serverseitigem JavaScript (SSJS) von Netscape oder den Active-Server-Pages (ASP) von Microsoft. JSP bauen auf Java Servlets auf und erlauben wie die beiden anderen Varianten das Mixen von HTML-Inhalt und Programmanweisungen. Als Standard benutzten JPS Java als Programmiersprache, allerdings werden auch eine Reihe von Tags für Java-freies Scripting bereitgestellt. Die Java-Server-Pages basieren auf einem Komponentenmodell, in dem JavaBeans und

[2] Hypertext Preprocessor: http://www.php.net/
[3] NeoWebServer: NeoSoft (http://www.neosoft.com)

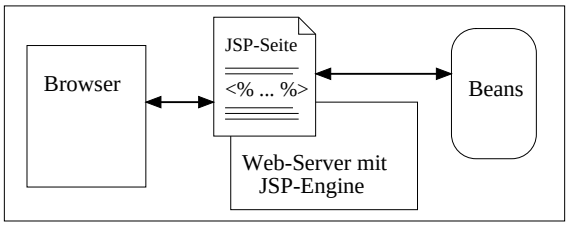


Abbildung 5.3: Java Server Pages

Enterprise JavaBeans (EJB) die Programmierung unterstützen. Wie in Abbildung 5.3 dargestellt, wird eine Anfrage direkt an eine JSP-Seite gesandt. Der JSP-Code kontrolliert Interaktionen mit JavaBeans-Komponenten und erzeugt eine dynamisch Web-Seite.

Als technologische Grundlage der Web-Erweiterung wurde das CGI-Konzept und die Implementationssprache Tcl ausgewählt. Für die Wahl von Tcl war neben den aufgeführten Vorteilen auch die in Abschnitt 5.2.1 angeführte Betrachtung verfügbarer XML-Software ausschlaggebend. Weiterhin existieren Tcl-Erweiterungen, die die CGI-Programmierung wesentlich vereinfachen. Das CGI-Konzept hat einen erwähnten Performancenachteil gegenüber Servermodulen, wie dem ausführlich betrachtetem Apache-Modul `mod_dtcl`. [ModTcl] Dieser liegt im Durchschnitt bei etwa 20%, ist aber bei den heute üblichen und im Rechenzentrum zur Verfügung stehenden Rechern (ca. 750 MHz Taktfrequenz) aus Nutzersicht kaum zu erkennen. Dieses Modul stellt auf Grund des vorhandenen Performancevorteils dennoch eine interessante Alternative dar, befindet sich aber noch in der Entwicklung und wurde wegen fehlender Funktionalitäten (z.B. Cookies, HTTP-Header, File-Upload, Mehrfachauswahllisten) nicht zur Implementation verwendet.

5.1.2 Entwurf des eOffice

Der Entwurf des eOffice als Softwareprodukt unterliegt einigen Voraussetzungen. Aus Nutzersicht muss es einfach erkennbar die Aufgabe einer Applikation zur Dokumentenerstellung erfüllen. Dabei sollten standardisierte Arbeitsschritte beibehalten werden, um ein einfaches Wechseln von einem Textsatzsystem zum eOffice zu gewährleisteten. Die Herausforderung aus Systemsicht liegt darin, dass Papierformulare und -dokumente nicht einfach übernommen elektronisch am Bildschirm angezeigt werden können und Formulare zur Online-Benutzung nicht als druckfähige Geschäftsdokumente akzeptiert werden. Um die Daten und die Darstellung von Dokumenten zu trennen, die in verschiedenen Formaten generiert werden sollen (z.B. für Druck oder Web-Betrachtung), eignet sich besonders die in Abschnitt 3.1 vorgestellte Meta-Sprache XML.

Die Anforderungen an das System werden aus Nutzersicht und aus der Sichtweise der Implementation dargestellt und nachfolgend ihre Konzeption erläutert. Außer einer anwenderfreundlichen Schnittstelle, die sich an bekannten grafischen Textsatzsystemen orientieren soll (Speichern, Öffnen, usw.) sind aus Nutzersicht weitere Aspekt ausschlaggebend. Der Nutzer benötigt die angeführten Grundfunktionalitäten in Bezug auf eine Online-Erstellung von Geschäftsdokumenten:

- Erstellen und Bearbeiten von Dokumenten
- Voransicht des aktuellen Dokumentes zur schnellen Gesamtansicht
- Erzeugung eines druckfertigen Dokumentes

Für den Aufbau und die Konzeption des Gesamtsystem treten weitere Schwerpunkte in den Vordergrund, die das System als Erweiterung der Intranet-Funktionalität des Universitätsrechenzentrums verstehen.

- Koppelung des Eingabeformulars mit der internen Datenverarbeitung

- Ständige Erweiterbarkeit um neue Geschäftdokumente
- Zugriff auf Datenbestände und Speicherkonzept
- Session-Management und Lock-Konzept
- Integritätssicherung der Dokumente

Unter Beachtung dieser Voraussetzungen wurde ein System konzipiert, dessen Schwerpunkte nachfolgend angeführt werden. Die Abbildung 5.4 zeigt den Entwurf der Architektur des eOffice auf Grundlage der Datenverarbeitung durch ein CGI-Programm.

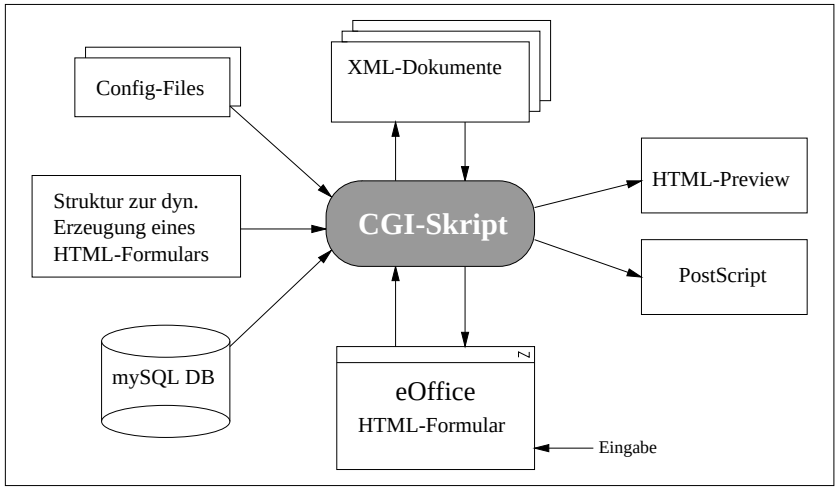


Abbildung 5.4: Entwurf der eOffice-Architektur

Dokumentenerstellung und Datenverarbeitung

Die Dokumentenerstellung und die interne Datenverarbeitung sind zwar in den Auflistungen der Voraussetzungen getrennt aufgeführt, gehören aber im Konzept zusammen und werden als Einheit entworfen. Sie bilden die Hauptbestandteile des in Abbildung 5.4 dargestellten Entwurfs. Während sich die interne Verarbeitung von Daten mit dem Einbinden dieser in XML-Strukturen beschäftigt, umfasst die Dokumentenerstellung alle drei aufgeführten Punkte der Nutzersicht. Das Erstellen und Bearbeiten von Dokumenten erfolgt durch die Eingabe von Daten in HTML-Formulare im Web-Interface des eOffice. Diese Formulare werden vom System dynamisch generiert und im Browser zur Anzeige gebracht. Sie unterliegen definierten Strukturen, welche den Inhaltstypen der analysierten Geschäftsdokumente entsprechen. Die Erstellung einer solchen Struktur wird in Kapitel 5.4 angesprochen. Nach dem Tätigen aller notwendigen Eingaben im Web-Formular löst der Nutzer die Übertragung der Formulardaten zum CGI-Programm aus. Alle Eingabedaten werden durch eine XML-basierte interne Verarbeitung in XML-Strukturen eingegliedert und weiterverarbeitet. In Abschnitt 3.3 wurden ausschlaggebende Gründe aufgeführt, die für das Document Object Model als Verarbeitungstechnologie für XML-Strukturen sprechen und deshalb eingesetzt wird. Fertige Dokumente liegen als XML-Dateien vor und können wieder in das beschriebenen Schema integriert werden, was einen Wiederbearbeitung ermöglicht. Somit sind Daten von Darstellungsanweisungen getrennt, was die Flexibilität der Dokumente erhöht. Diese können in verschiedene Darstellungsformate konvertiert werden. Als Voransicht eignet sich ein HTML-Preview, der schnell die annähernd äquivalente Gestalt eines druckfähigen Geschäftsdokumentes anzeigt und dem Nutzer eine Gesamtansicht des erstellten Dokumentes bietet. Wie oben erwähnt, kann diese ein HTML-Formular nur sehr beschränkt vermitteln. Als

Endprodukt entsteht eine druckfähige Version des Geschäftsdokumentes. Da PostScript das allgemein benutzte Format zum Drucken im URZ ist, liegt die Erzeugung von druckfähigen Dokumenten in PostScript nahe. Zur Umwandlung der XML-Dateien ist ein weiteres System oder weitere unterstützende Software nötig. Techniken zur Konvertierung von XML-Datei wurden in Abschnitt 3.4 aufgeführt und werden in Abschnitt 5.2.1 auf ihre Verwendbarkeit betrachtet.

Erweiterbarkeit

In der Arbeit werden sieben Dokumententypen analysiert, die durch das System erstellt werden sollen. Um die Möglichkeit zu bieten, Dokumente weiterer Typen zu erstellen, wird das System für neue Dokumententypen erweiterbar konzipiert. Diese Erweiterbarkeit des Systems hängt stark mit der Erzeugung von HTML-Formularen zusammen. Das CGI-Skript generiert dynamische HTML-Formulare, die einer definierten Struktur unterliegen. Mit der Definition neuer Formularstrukturen, die dem Aufbau und den Inhaltstypen weiterer Geschäftsdokumente entsprechen, können somit vom System auch neue Formulare erzeugt werden, die die Erstellung neuer Dokumente erlauben. Aus diesem Grund muss eine geeignete Struktur zur Erzeugung von Formularen bereitgestellt werden, die das System sinnvoll umsetzen kann. Diese kann aus einer Dokumenttyp-Definition eines Geschäftsdokumentes abgeleitet werden. In der weiteren Arbeit wird diese Struktur zur dynamischen Erzeugung eines HTML-Formulars auch kurz mit Formularstruktur bezeichnet. Die zur Zeit der Dokumentenerstellung statisch verfügbare Formularstruktur wurde einer Alternative vorgezogen, bei der diese Struktur erst zum Programmablauf erzeugt wird, was aber einer erhöhten Rechenzeit während der Abarbeitung entspräche. Zu erstellende Dokumententypen werden in einem XML-Format gespeichert und unterliegen einer festen Definition eines Dokumententyps, also einer Auszeichnung von Inhaltstypen. Hierbei liegt es nahe, sie auch als die Struktur zur Erzeugung von HTML-Formularen zu verwenden. Dieses Konzept wird auf Grund seiner Komplexität separat in Abschnitt 5.4 erläutert.

Zugriff auf Datenbestände

Der Zugriff auf Datenbestände bezieht sich auf alle Daten, die das CGI-Programm während der Ausführung lesen oder schreiben muss. Nach dem Erzeugen werden in XML strukturierte Geschäftsdokumente gespeichert, wofür zwei Möglichkeiten der Ablage in Betracht gezogen werden können. Zum einen ist das Speichern der Dokumente in einer Datenbank denkbar und zum anderen ist das Sichern der Dokumente als Dateien wie herkömmlich praktikabel. In letzter Zeit wurden XML-fähige Datenbanksysteme entwickelt, die aber für den konkreten Einsatz keinen Vorteil bringen und sich eher für andere Zwecke eignen, z.B. zur Ablage und ständigen Aktualisierung von XML-strukturierten Transaktionsdaten. Da die Dateien auch schnell von Hand einsehbar und programmtechnisch einfach handhabbar sind, werden diese bevorzugt und somit die Geschäftsdokumente als XML-Dokumente im Dateisystem abgelegt. Das XML-Format ist auch anderweitig einsetzbar, z.B. zur Erstellung eines Mitarbeiterverzeichnis oder als Datenbankauszug. Diese XML-Files müssen für den Web-Server ebenso les- und schreibbar sein. Weitere zur internen Verarbeitung, Konfiguration oder Konvertierung notwendigen Dateien werden dem Web-Server mit einem Leserecht zur Verfügung gestellt.

Zugriff auf Datenbestände impliziert auch den Zugriff auf die Datenbank der Firmenadressen, die als MySQL-Datenbank vorliegt. Um Adressdaten abzufragen, wird ein Nutzer mit lesendem Zugriff benötigt. Als Alternative zur Abfrage der Adressdatenbank während der Verarbeitung besteht die Möglichkeit, eine XML-Struktur von Adressdaten anstelle der Datenbank auszulesen. Bei der CGI-Verarbeitung und Anfrage nach Adressdaten könnte dann das Adress-XML-File angesprochen werden, ohne die Datenbank zu kontaktieren. Der direkte Datenbankzugriff ist durch entsprechende Tcl-Erweiterungen unkompliziert, braucht aber eine etwas längere Zugriffszeit. Durch eine XML-Adress-Struktur wird die Zugriffszeit minimiert, die während der Bearbeitung zum Auslesen der Adressen benötigt wird. Die Datenbank wird allerdings nur einmal pro Dokumentenerstellung benötigt, womit dieser Zugriff die Erstellung nicht bedeutend verzögert. Um eine XML-Datei für Adressen zu erzeugen, wird der Administrator eine Datenbankanfrage ver-

anlassen. Die Adress-Datenbank befindet sich in ständiger Wartung und wird derzeit häufig aktualisiert. Nach jeder Aktualisierung müsste der Administrator des Systems das XML-Adress-File neu anlegen. Um dies zu umgehen, wird der Datenbankzugriff zur Erstellungszeit bevorzugt. Die Möglichkeit des Zugriffs auf ein Adress-XML-File bleibt aber interessant, zumal im XML-File statt der englischen Begriffe für einzelne Einträge aus der Datenbank dann besser deutsche Begriffe stehen könnten, die in die Formularerzeugung einfließen.

Speicherkonzept

Nachdem im Zugriff auf Datenbestände schon auf die Art der zu sichernden Daten eingegangen wurde, befasst sich das Speicherkonzept mit dem Ort zur Aufbewahrung von Dateien und den Namenskonventionen für die Geschäftsdokumente. Alle Dateien können lokal auf dem Computer gelagert werden, auf dem auch das System in Betrieb genommen wird. Bei entsprechender Rechtevergabe ist das Ablegen der Daten auch in dem an der Universität üblichen AFS möglich.

Um das Speichern von Dateien in falschen Verzeichnissen zu vermeiden, muss vom System ein einheitlicher Pfad festgelegt werden. Hierbei werden die aktuellen Verzeichnisstrukturen im Büro übernommen:

'Pfad für Dokumente'/'Dokumententyp'/'Erstellungsjahr'/'Dokumentenname'.

Der Dokumentenname unterliegt folgender Konvention:

'$\leq$ 15 benutzerdefinierte alphanummerische Zeichen'-'Nutzerkennzeichen'-'Datum'-'Zeit'.xml.

Des Weiteren werden Lock-Dateien und Signatur-Dateien erstellt, die sich nur in der Dateiendung unterscheiden. Dazu muss ein Verzeichnis für die Systemdateien des eOffice vorgesehen werden.

Session-Management und Locking

Da das zu entwickelnde System von mehreren Nutzern gleichzeitig verwendet werden kann, es sich also um eine Multi-User-Anwendung handelt, ist das Führen von Sitzungen pro Nutzer notwendig. Um unterschiedliche Sessions zu erlauben und Nutzer auf einen Mehrfachzugriff auf ein Dokument hinzuweisen, wurde ein Session- und Locking-Konzept aufgestellt. Für ein geeignetes Management der Nutzer-Sessions wurden zwei Varianten betrachtet. Ein eindeutiger Identifikator repräsentiert eine Session und wird zur Generierung eines Dateinamen verwendet. Zur Sicherung aller notwendiger Daten einer Sitzung kann eine Datei mit dem entsprechenden Dateinamen erzeugt werden, die meist Variablen für Rechte oder Zustände enthält. Dieser Identifikator kann in einem Cookie [Kri97] gespeichert werden, der einen Namen und einen Wert besitzt. In einer möglichen Variante existiert der Cookie solange die Browser-Sitzung aktiv ist, d.h. wenn der Browser geschlossen wird, verfällt auch der Cookie. Falls ein Nutzer zwei Sessions in einer Browser-Sitzung startet, wird der Cookie mit dem Id der zweiten Session überschrieben, weshalb diese Methode nicht zum Einsatz kommen kann. Deshalb wird die in CGI's übliche Technik verwendet, den Sitzungsidentifikator in jedem Request als versteckte Variable zu übergeben. Für jede neue Sitzung wird für einen authentifizierten Nutzer ein neuer Id und das Session-File erzeugt. Die Authentifikation kann durch das schon erwähnte .htaccess-Konzept von Apache geschehen.

Da pro Session nur ein Dokument bearbeitet werden soll, kann während der internen Verarbeitung ein durch den Session-Id gekennzeichnetes Lock-File für diese Dokument erstellt werden. Andere Nutzer, die auf dieses Dokument zugreifen wollen, werden auf den Status des momentan in Bearbeitung befindlichen Dokumentes hingewiesen. Diese Technik ähnelt die dem Lock-Konzept des `vi` und von `netscape` und ist für das eOffice ausreichend. Andere betrachtete Lock-Konzepte sind aufwendiger und nicht notwendig. Ein Nutzer sollte das fertig bearbeitete Dokument ordnungsgemäß schließen. Andernfalls bleibt das Lock-File vorerst erhalten; wird aber nach einer definierten Zeit gelöscht.

Integritätssicherung der Dokumente

Im Rechenzentrum besteht kein Risiko in Bezug auf die Sicherheit der abgelegten Dokumente. Das Anlegen von Zugriffsberechtigungen für die Mitarbeiter zur Erzeugung und Bearbeitung von Geschäftsdokumenten wurde in Abschnitt 2.1.1 angesprochen. In einer Nutzerverwaltung können diese für jeden Mitarbeiter und jeden Dokumententyp definiert und somit Benutzergruppen gebildet werden. Diese Gruppen spiegeln den Aufgabenbereich des jeweiligen Nutzers wider und sind aber auch für weitere Konzepte des Einbeziehens URZ-fremder Mitarbeiter der TU verwendbar. Ein Nutzer kann Zugriffsrechte für jeden Dokumententyp erhalten, die durch den Administrator des Systems erteilt werden. Authentifizierte Nutzer werden Dokumente von anderen Mitarbeitern nicht ungefragt verändern; Versehen dieser Art könnten aber vorgebeugt werden. Es gibt auch andere Gründe, warum ein Konzept zur Sicherung der Integrität der Geschäftsdokumente durchaus interessant ist. Falls ein Dokument in Gemeinschaftsarbeit geschrieben wird, sind der aktuelle Bearbeitungstand und der Name des letzten Editors von Bedeutung. Im Abschnitt 4.5 wurde die Digitale Unterschrift vorgestellt, die neben der Unversehrtheit auch die Echtheit und Glaubwürdigkeit der Daten bestätigen kann. Sie wird derzeit allerdings bei keinem der Dokumente, die die Universität verlassen, die handschriftliche Signatur ersetzen. Diese Dokumente müssen in Papierform vorliegen und fordern in der Regel eine handschriftliche Unterschrift. Bis der gesamte Geschäftsverkehr elektronisch über das Internet abgewickelt wird, werden noch einige Jahre vergehen. Eine digitale Signatur erreicht nur ihr Ziel, wenn das Dokument, das sie unterschreibt, auch in digitaler Form beim Geschäftspartner vorliegt. Im Bereich der Universität ergeben sich durch ein Freischalten des eOffice für einen bestimmten Kreis von Personen außerhalb des URZ Ansatzpunkte für Überlegungen in dieser Richtung. TU-Mitarbeiter könnten die internen Dokumente über das eOffice lesen, deren Korrektheit durch die Signatur prüfen und den Papieraufwand zum Versenden der Papier-Dokumente einsparen.

Um den Aufwand für die Benutzer zu verringern, wird ein Verfahren zur digitalen Signierung ausreichen, das auf einem Schlüsselpaar basiert und vom Web-Server ausgeführt wird. Dieser unterschreibt ein Dokument für einen authentifizierten Nutzer und erstellt ein Signatur-File, das die Signatur und den Nutzer enthält. Somit kann die Integrität nachgewiesen und der Nutzer, der das Unterschreiben veranlasst hat, bestimmt werden. Andere Verfahren, wie das Generieren eines Schlüsselpaars für jeden Nutzer, sind zwar sicherer aber auch um einiges aufwendiger und machen dem Nutzer zusätzliche Arbeit. Für die dargestellte Situation innerhalb des Rechenzentrums genügt die einfache Methode. Falls das Konzept in der Weiterentwicklung des Systems nicht mehr ausreicht, kann ein neues Verfahren ein anderes strukturiertes Signatur-File erstellen, ohne an der Struktur des Geschäftsdokumentes etwas ändern zu müssen.

5.2 Nutzbare Software

5.2.1 XML-Software

Die Web-Seite `http://www.xmlsoftware.com` bietet eine reichhaltige Auswahl an XML-Software, auf der auf die folgenden für den Einsatz betrachteten Produkte verwiesen wird. Die Auswahl beschränkt sich auf freie Software.

XML Parser

Da die Verwendung des DOM vorgesehen ist, wird neben der Verfügbarkeit eines XML Parsers auch auf DOM-Fähigkeit Wert gelegt. Das Level 1 des Document Object Model genügt für die interne Verarbeitung des eOffice, weil es alle benötigten Funktionen zur Manipulation von XML-Dokumenten abdeckt und meist als einziger Level in den betrachteten Werkzeugen realisiert ist. Die Verfügbarkeit des oder der notwendigen Softwarepakete ist Voraussetzung für die Verwendung einer Programmiersprache, da die Verarbeitung von XML-Strukturen die Grundlage des System darstellt.

- Als Alternative zu CGI wurden Server-seitige Skriptsprachen wie PHP[4] erwähnt. Der PHP XML Parser parst willkürliche XML-Daten und erzeugt ein Feld der Struktur aller Tag- und Datenelemente. Da es für PHP keine DOM-Anbindung gibt, entfällt der Hypertext-Prozessor als Implementationssprache.

- Die XML-Implementierungen für Tcl erfüllen die gestellten Erwartungen und bestätigen Tcl als Programmiersprache für das CGI. Sie liegen in den Versionen TclXML 1.2 (Parser) und TclDOM 1.6 vor und wurden von Zveno Pty Ltd[5] entwickelt.

- Perl ist die primäre CGI-Skriptsprache und verfügt ebenso wie Tcl über XML-Erweiterungen, die die gestellten Erwartungen erfüllen. Die als Perl-Module[6] implementierten XML::Parser und XML::DOM sind schon praxisbewährt und für eine einfache Verarbeitung prädestiniert.

- Weitere interessante Produkte liegen in Java vor, z.B. bieten sich auf diesem Gebiet 'Xerces'[7] des Apache XML Projekts und 'XML parser for Java'[8] von Oracle an. Da mit den XML-Implementationen in Tcl schon eine Vorentscheidung auf Tcl gefallen ist, spielen diese fast keine Rolle mehr.

XSLT Engines

XSLT Engines wandeln XML-Dokumente unter Verwendung von Stylesheets in Dokumente anderer Struktur um. Diese Werkzeuge bieten einen Ansatz, um die in XML-Struktur vorliegenden Geschäftsdokumente in Darstellungsformate zu konvertieren.

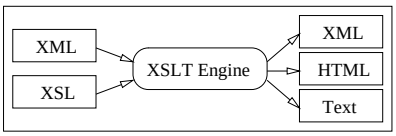


Abbildung 5.5: XSLT Engine

Dazu zählen wiederum XML, HTML und Text. Eine Textkonvertierung kann damit auch LaTeX-Dokumente erzeugen, womit das Generieren der PostScript-Dokumente durch LaTeX und das Paket `dvips` realisiert werden können. Die Bedeutung von XSLT ist in Abschnitt 3.4 erläutert worden.

Da für Tcl keine XSLT-Implementation vorliegt, werden nachfolgend unterschiedliche Konvertierungswerkzeuge betrachtet. Bei der Auswahl der XSLT Engine wurde besonders auf eine einfache Integration der Engine (Java) in das eOffice Wert gelegt. Das geschieht beispielsweise durch die Benutzung eines Kommandozeilenaufruf, wodurch eine leichte Austauschbarkeit der Produkte unterstützt wird. Als Folge der rasanten Entwicklung in diesem Gebiet ist in absehbarer Zeit auch für Tcl mit einer XSLT-Erweiterung zu rechnen.

- LotusXSL[9] ist eine XSL Transformation Engine von IBM. Sie basiert auf Xalan, einer Engine des Apache-XML-Projekts und ist demnach in Java implementiert.

- Xalan[10] ist ein XSL-Prozessor aus dem Apache-XML-Projekt, der XML in HTML, einfachen Text oder andere XML-Formate umwandelt. Diese Java-Engine hat sich als sehr stabil erwiesen und wird oft eingesetzt.

- Das Saxon-Paket[11] ist eine Sammlung von Tools zum Verarbeiten von XML Dokumenten. Es implementiert XSLT 1.0 und XPath 1.0 sowie eine Reihe weiterer Erweiterungen und ist damit mächtiger als einfache XSLT-Implementierungen.

[4] PHP, Hypertext Prozessor: http://www.php.net
[5] Zveno Pty Ltd, XML-Pakete: http://www.zveno.com/zm.cgi/in-tclxml/
[6] C.Cooper: http://wwwx.netheaven.com/~coopercc/xmlparser/intro.html
[7] Apache XML Project: http://xml.apache.org/xerces-j/index.html
[8] Oracle: http://technet.oracle.com/tech/xml/parser_java2/
[9] LotusXSL: http://www.alphaworks.ibm.com/tech/LotusXSL
[10] Xalan-J: http://xml.apache.org/xalan-j/index.html
[11] Saxon: http://users.iclway.co.uk/mhkay/saxon/index.html

- XT[12] ist in Java entwickelt und implementiert XSLT 1.0. Die verfügbaren Kommandozeilenaufrufe zur Konvertierung gleichen denen von Xalan. XT wird meist mit dem XML Parser XP verwendet. Beide Produkte stammen ursprünglich von James Clark.

XSL Formatters

Wie in Abschnitt 3.4 angedeutet, gibt es zwei XSL-Technologien, um Daten in verschiedene Formate zu transformieren. Nachfolgend werden drei betrachtete XSL Formatierer vorgestellt, die auf Formating Objects basieren.

- FOP[13] ist eine Java-Anwendung, die im Rahmen des Apache XML Projektes entwickelt wird. Sie erzeugt PDF-Dokumente und verwendet zur Formatierung des Dokumentes wahlweise DOM oder SAX.
- PassiveTeX[14] ist ein TEX-Paket, das XSL:FO direkt nach PDF formatiert. Es verwendet die XML Parser XT und xmltex.
- Das Paket TeXML[15] von IBM basiert wiederum auf Java. Es erstellt aus einem XSL-Stylesheet und einem XML-Dokument das Zwischenformat TeXML, welches durch ein weiteres Java-Programm nach TEX transformiert wird. Im Test bestätigte das Tool seine Funktionalität, benötigt aber zwei Stylesheets. Der Einsatz des Tool ist nicht nötig, da auch aus XML-Dokumenten und einem XSL-Stylesheet mittels einer XSLT Engine LATEX generiert werden kann.

Publishing Systeme

Die Konvertierung nach HTML und PostScript könnte auch einem der wenigen freien Publishing-Systeme überlassen werden. Da diese Systeme kein PostScript unterstützen und schlecht mit der internen Datenverarbeitung des eOffice zu koppeln sind, passen sie nicht in das Konzept des eOffice und werden nur kurz erwähnt.

- Aus der Arbeit des Apache-XML-Projekt stammt das auf Java basierende System Cocoon[16], das Technologien, wie DOM, XML und XSL verwendet, um Web-Dokumente zu erzeugen. Cocoon braucht Pakete, wie Xerces, Xalan und FOP, um aus XML-Dateien Formate, wie PDF, XHTML zu erstellen.
- Resin[17] ist ein System, welches auf Java Server Pages basiert und zum Zwecke der Erzeugung von Web-Seiten XSLT verwendet. Es enthält einen XML-DOM-Parser, aber auch mehrere Zusatztools zur Webseitengestaltung und ist sehr komplex.

Eine interessante Erweiterung ist außerdem das Modul für XSLT `mod_xslt`[18] für Apache. Beim Anfordern eines URL wird die spezifizierte Datei aus einer XML-Datei und einem XSL-Stylesheet generiert, der im DOCTYPE angeführt ist und abhängig von der angeforderte Datei-Extension ausgewählt wird.

5.2.2 Auswahl

Folgende Produkte werden bei der Implementation verwendet:

- Tcl 8.2, `http://www.scriptics.com/`

[12] XT: http://www.jclark.com/xml/xt.html
[13] FOP: http://xml.apache.org/fop/
[14] PassiveTeX: http://users.ox.ac.uk/ rahtz/passivetex/
[15] TeXML: http://www.alphaworks.ibm.com/tech/texml
[16] Cocoon: http://xml.apache.org/cocoon/
[17] Resin: http://www.caucho.com/index.xtp
[18] Apache Module for XSLT: http://modxslt.userworld.com/

- TclXML 1.2, `http://www.zveno.com/zm.cgi/in-tclxml/`
- TclDOM 1.6, `http://www.zveno.com/zm.cgi/in-tclxml/in-tcldom/`
- XSLT Engine Xalan 1.0.1, `http://xml.apache.org/xalan/index.html`
- XML Parser Xerces 1.0.3, `http://xml.apache.org/xerces-j/index.html`
- MySQLTcl 1.53, `http://www.xdobry.de/mysqltcl/`
- GnuPG 1.0.2, `http://www.gnupg.org/download.html`

Tcl, TclXML, TclDOM

Auf Grund der einfachen und flexiblen Syntax von Tcl und den zur XML-basierten internen Verarbeitung vorhandenen Pakete TclXML und TclDOM wurde Tcl als die Implementierungssprache für CGI-Programme ausgewählt. Die Arbeit mit TclDOM, welches das DOM Level 1 an die Skriptsprache bindet, fordert mindestens die Tcl-Version 8.2. Die Pakete TclXML und TclDOM gewährleisten zusammen eine auf XML basierende Manipulation von XML-Strukturen und werden in detaillierter Form in Abschnitt 6.1.2 angesprochen.

Xalan/Xerces

Da momentan keine XSLT-Engine für Tcl existiert, wird ein einfach zu integrierender Transformator zur Konvertierung der XML-Dokumente in darstellbare Formate verwendet, der nachweislich gut funktioniert. Die Auswahl fiel auf Xalan 1.0.1, der ein Teil des Apache-XML-Projektes ist und meist zusammen mit dem XML-Parser Xerces verwendet wird. Diese Produkte wurden gewählt, da erste eigene Tests mit den Werkzeugen positiv verliefen. Außerdem werden sie sehr oft auch als Grundlage von Implementationen anderer Anbieter genutzt und gelten als stabil. Die XSLT Engine ist Open Source und basiert komplett auf Java. Sie implementiert die W3C Empfehlung XSL Transformations (XSLT) Version 1.0 vom 16.November 1999 und die XML Path Language (XPath) Version 1.0 vollständig. Xalan kann zum Parsen von XML-Dokumenten und XSL Stylesheets optional ein eigens implementiertes performancesteigerndes Document Table Model (DTM) verwenden, ein „pseudo-DOM“, das mit Integer-Feldern arbeitet. Zum Parsen der Dokumente wird automatisch der Xerces-Java Parser benutzt, welcher aber auch durch andere Parser ersetzt werden kann. Da kein Grund besteht, einen anderen Parser einzusetzten, wird Xerces in der Version 1.0.3 verwendet. Voraussetzung der Verwendung dieser Pakete ist die Installation von Java.

Mysqltcl

Für den Zugriff auf Firmenadressen, die in einer MySQL-Adressdatenbank aufbewahrt werden, ist ein Anwendung nötig, die die Fähigkeiten von Tcl erweitert. Mysqltcl ist eine solche Erweiterung der Tool Command Language (Tcl), die einen einfachen Zugriff auf MySQL Datenbank-Server erlaubt. Das Interface kann in zwei Varianten verwendet werden, einerseits als statisch gelinkte Anwendung (mysqltclsh oder mysqlwish) oder als dynamisch gelinkte Shared Library.

Digitale Signatur - GnuPG

Zur Erzeugung digitaler Signaturen wurde der GNU Privacy Guard [GnuPG] ausgewählt, einer freien Implementierung des vorgeschlagenen Internet-Standards OpenPGP (RFC2440). GnuPG ist ein Werkzeug für sichere Kommunikation und enthält auch Funktionen zum Signieren und Verifizieren von Dokumenten. Der Privacy Guard ist ein vollständiger und freier Ersatz für PGP, der ohne Einschränkungen auf Grund des Verzichtes auf IDEA und RSA verwendet werden kann.

Eine weitere frei erhältliche Alternative ist DSig[19], ein Java-Paket des W3C, das die von der XML Signature Working Group erarbeitete XML-Signature unterstützt. GnuPG wurde bevorzugt, da es sich als das performantere Werkzeug erwies.

5.3 Entwurf der Nutzerschnittstellen

Um dem Nutzer ein übersichtliches und anwenderfreundliches Nutzer-Interface zu bieten, wurde eine einfache Schnittstelle (Abbildung 5.6) konzipiert, die Grundlage aller weiteren Eingabemasken ist. Durch ein aus Buttons bestehendes Menü können ausführbare Funktionalitäten sofort überblickt werden. Die Struktur ist Grundlage aller Interfaces des eOffice und wird mittig im Browser dargestellt. Im Rechenzentrum wird der Browser Netscape unter Linux und Windows NT verwendet. Die zwischen beiden Systemen auftretenden Unterschiede in der Darstellung der Web-Formulare (z.B. die Differenzen in der Darstellung der Buttons) sind minimal und können vernachlässigt werden.

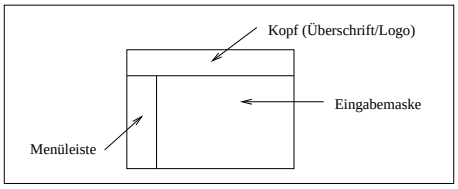


Abbildung 5.6: Aufbau aller Schnittstellen

Da die Authentifizierung eines Nutzers nicht direkt im eOffice, sondern über den .htaccess-Mechanismus von Apache erfolgt, ist dafür kein Interface vorgesehen. Der Entwurf sieht drei Bereiche vor, in denen der Nutzer agieren kann. Diese sind Hauptbereich, Arbeitsbereich und Administratorbereich, die jeweils ein eigenes für jeden Bereich festes Menü erhalten. Die Interfaces dieser Bereiche sind demnach aus dem allgemeinen Kopf/Logos, dem jeweiligen Menü und einer variablen Texteingabemaske aufgebaut. (Abbildung 5.6) Sie werden auch als Hauptbildschirm, Arbeitsbildschirm und Administratorbildschirm bezeichnet. Durch ihr Menü werden verschiedene zugehörige Eingabemasken und Funktionen ausgewählt. Das Menü ist für jede mögliche Eingabemaske in einem dieser Bildschirme gleich. Die Struktur der Navigation zwischen den Interfaces ist in Abbildung 5.7 dargestellt.

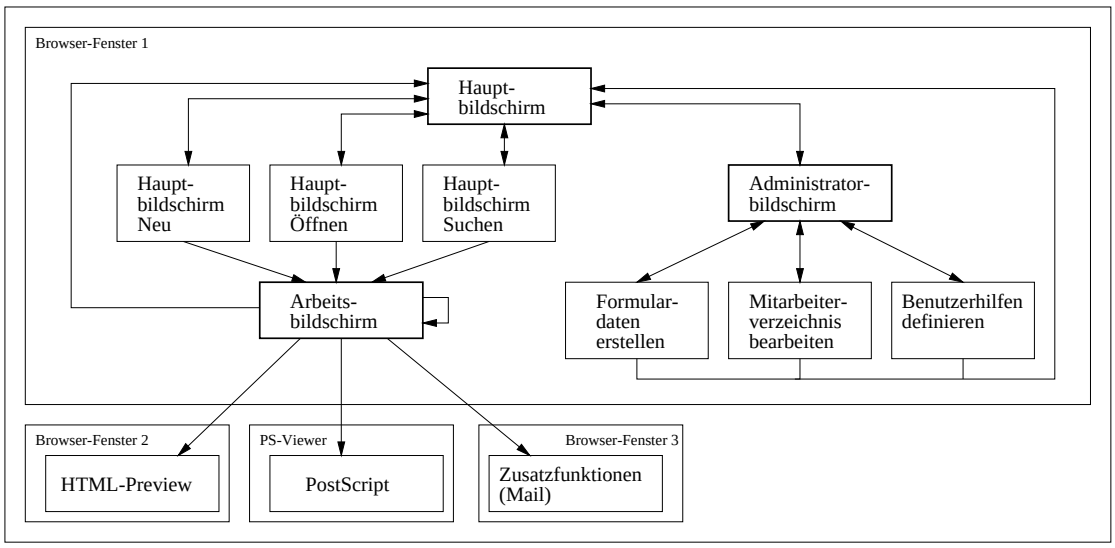
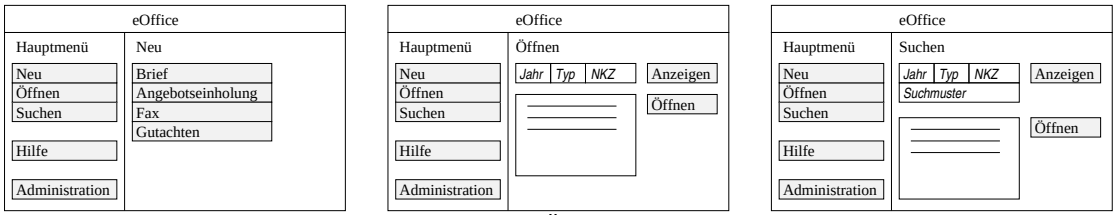


Abbildung 5.7: Navigation zwischen den Nutzerschnittstellen

Alle Schnittstellen sind symbolisch eingerahmt, womit ein Browser-Fenster oder ein externer

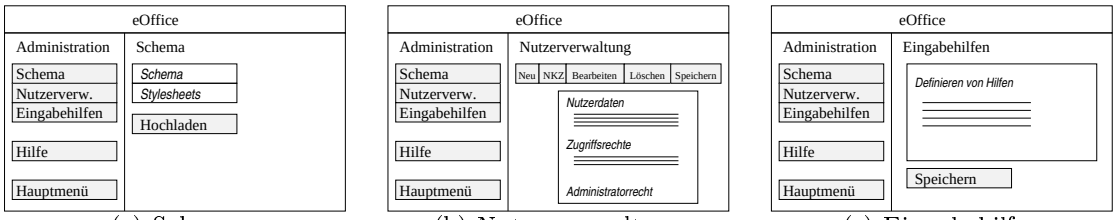
[19] DSig: http://www.w3.org/PICS/refcode/DSig

Viewer verdeutlicht wird. Eine Navigation findet nur im Browser-Fenster 1 statt, in dem der Nutzer die in der Abbildung 5.7 aufgezeigten Arbeitsschritte ausführen kann. Im Hauptbildschirm können durch das Hauptmenü Eingabemasken für das Neuerstellen, Öffnen und Suchen von Dokumenten angefordert werden. Das Laden und Neuerstellen eines Geschäftsdokumentes geschieht immer im Arbeitsbildschirm, wohin die aufgeführten Funktionen des Hauptmenüs führen. Durch das Schließen eines Dokumentes gelangt der Nutzer wieder zum Hauptbildschirm zurück. Im Arbeitsbildschirm gibt es viele Funktionen, die weitere Interaktionen mit dem Nutzer fordern, z.B. *Speichern als, Signieren*, usw. (siehe Abbildung 5.10). Das Resultat der vom Nutzer gewählten Funktionalität wird wenn nötig im Fenster selbst bestätigt, d.h. über dem Eingabeformular muss Platz dafür in Anspruch genommen werden. Diese Bestätigung einer Funktionsausführung lässt sich geeignet in das Interface integrieren und ist für den Nutzer gut einzusehen. Eine andere Möglichkeit, ein weiteres Fenster für das Resultat zu öffnen (das der Nutzer immer wieder schließen muss), wird daher nicht verwendet.



(a) Neu (b) Öffnen (c) Suchen

Abbildung 5.8: Interface: Hauptmenü und Funktionen



(a) Schema (b) Nutzerverwaltung (c) Eingabehilfen

Abbildung 5.9: Interface: Administratormenü und Funktionen

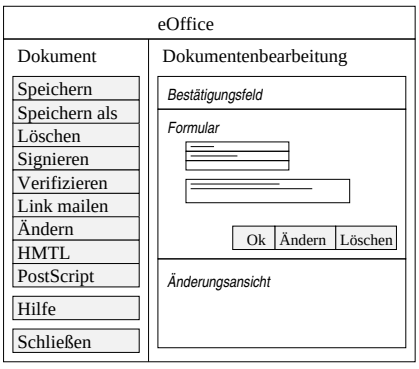


Abbildung 5.10: Interface: Dokumentenerstellung

Die Entwürfe der Interfaces sind in den Abbildungen 5.8 bis 5.10 dargestellt. Darin sind Buttons leicht grau unterlegt. Die Abbilungen 5.8 und 5.9 stellen die verschiedenen Eingabemasken des Haupt- und Administratorbereiches dar, die durch Betätigen der entsprechenden Buttons der Menüleisten angezeigt werden können. Zur Anzeige einer Hilfe-Seite ist in allen Interfaces ein Button „Hilfe“ enthalten.

In Abbildung 5.8/(a)Neu werden verfügbare Dokumententypen als Buttons aufgeführt, um sie zur Neuerstellung zu selektieren. Die Maske „Öffnen“ fordert Daten wie Nutzerkennzeichen, Dokumententyp und Erstellungsjahr, um eine Liste von Dokumenten anzuzeigen. Durch die Angabe eines zusätzlichen Textmusters kann eine Suche nach Dokumenten deren Inhalt diesem Muster entsprechen in „(c)Suchen“ veranlasst werden. In den Listen ist das Selektieren eines Dokumentennames zum Öffnen möglich. Abbildung 5.9/(a)Schema stellt die Schnittstelle dar, die dem Systembetreuer die Möglichkeit bietet, notwendige Datei zur

Integration neuer Geschäftsdokumente auf den Server zu laden. In der Maske „Nutzerverwaltung“ stehen Buttons und ein Eingabeformular zur Aktualisierung eines Mitarbeiterverzeichnisses zur Verfügung. Zur Änderung von Einträgen von Struktureinheiten bietet die Eingabemaske „(c)Eingabehilfen“ ein mehrzeiliges Eingabefeld.

Die Abbildung 5.10 zeigt das Interface der Dokumentenerstellung. Es beinhaltet Buttons, die an gewöhnliche Software zur Dokumentenerstellung erinnern (Speichern, Schließen, usw.). Des Weiteren sind Funktionalitäten wie Signieren und Verifizieren zum Erstellen und Prüfen von digitalen Signaturen vorhanden. Wie schon in Abbildung 5.7 dargestellt ist ein einfaches Mailformular (mit Angabe der Mailadresse, des Thema und Daten in den Textbereich) angedacht, durch das der Link des aktuell in Bearbeitung und befindlichen gesicherten Dokumentes versandt werden kann. Deshalb wurde der Button „Link versenden“ angegeben. Eine Änderungsansicht kann durch „Ändern“ ein- und ausgeblendet werden. Von den Buttons „HTML“ und „PostScript“ werden Mechanismen zur Visualisierung der bearbeiteten Dokumentes als HTML-Ansicht und PostScript-Darstellung aufgerufen. Im Eingabeformular befinden sich Buttons zur Navigation in und Aktualisierung des XML-Dokumentes.

5.4 Datenstruktur der XML-Dokumente

XML-Dokumente unterliegen einer definierten Struktur, die auch als XML-Anwendung bezeichnet wird. Diese beschreiben die Struktur aller Objekte der Anwendung und bestimmen deren Inhaltstypen. Um die Geschäftsdokumente in XML zu repräsentieren ist entweder das Schaffen von derartigen Strukturen oder die Verwendung existierender Typdefinitionen nötig. Wie in der Schlussfolgerung des Kapitels 4 ausgeführt, bietet sich keine betrachtete DTD als Dokumenttyp-Definition für die Geschäftsdokumente des URZ an. Aus diesem Grund werden eigene Strukturen entworfen, die den gestellten Anforderungen entsprechen.

Im Entwurf der Architektur des Gesamtsystems wurde darauf hingewiesen, dass eine optimale Typdefinition nicht nur als XML-Anwendung für die Geschäftsdokumente dienen soll, sondern gleichfalls als Struktur oder Muster einer Struktur zum Aufbau der HTML-Formulare. Auf Grund der erwähnten Vorteile von Schemata gegenüber DTD's werden sie bevorzugt. Allerdings sind Dokumenttyp-Definitionen auf den ersten Blick einfacher zu überschauen. Da Schemata die selbe Syntax wie XML-Dokumente verwenden, können sie ohne Nutzung zusätzlicher Parser-Software verarbeitet werden. Die Definition eigener Strukturen zum Aufbau von Formularen hat den weiteren Vorteil, dass die Tags deutsche Bezeichnungen erhalten und somit gleich im erzeugten Formular zur Namensgebung der Formulareingabefelder verwendet werden können. Eine Übersetzungstabelle für Elemente vorhandener kommerzieller Tags in deutsche Geschäftsprozessbezeichnungen ist aufwendig und dient nicht der Erweiterbarkeit des Systems. Dem Systembetreuer wird somit nicht zugemutet, sich zur Einbindung neuer Geschäftsdokumente aus verschiedenen DTD's oder Schemata eine geeignete Struktur zusammenstellen zu müssen und für jedes meist englische Tag eine deutsche, dem Geschäftsdokument entsprechende Übersetzung, festzulegen.

In Abschnitt 2.2.8 wurden Klassen von Geschäftsdokumententypen definiert, um die Integration verschiedener Dokumententypen zu vereinfachen. Dies wird erreicht, indem für die zu einer Klasse gehörenden Dokumente nur eine XML-Anwendung erforderlich ist, um deren Struktur und Inhalt zu repräsentieren. Des Weiteren ist auch nur ein Stylesheet für eine Darstellungsart notwendig, was die Arbeit zur Erstellung dieser beschleunigt. Allerdings wirft der Aspekt der Erzeugung von Formularstrukturen aus einer XML-Anwendung, die mehrere Geschäftsdokumenten repräsentiert, ein Problem auf. Die Funktion von DTD's und Schemata wurde am Beispiel des Dokumententyps letter in Kapitel 3 verdeutlicht. Dokumenttyp-Definitionen und Schemata normieren die Struktur und beschreiben die Inhalte von XML-Dokumententypen. Das bedeutet, die gebildeten Klassen von Geschäftsdokumententypen werden auf Dokumententypen im Sinne von XML reduziert. Im Zusammenhang mit der Erstellung von verschiedenen HTML-Formularen für jeden Geschäftsdokumententyp ist dies allerdings problematisch, wie das folgende Beispiel stellvertretend für ähnliche Situationen zeigt.

Die Angebotseinholung und das Fax gehören zu der Klasse Brief. Ein Fax enthält das Ele-

ment *Faxnummer des Empfängers*, welches im Typ Angebotseinholung nicht vorkommt. In einem Schema für die Klasse Brief würde der Eintrag für diese Faxnummer mit `<... minOccurs='0' maxOccurs='1'>` abgeschlossen. Somit kann die *Faxnummer* im Dokument im Falle eines Faxes vorhanden sein oder auch nicht im Falle der Angebotseinholung. Soll jetzt aber ein HTML-Formular aus diesem Schema generiert werden, so kann die zu verarbeitende Software nicht entscheiden, ob ein Formular für die Angebotseinholung oder das Fax erzeugt werden soll. In jedem Falle ist ein Eingabefeld *Faxnummer* in der Angebotseinholung überflüssig und würde nur zur Irritation der Nutzer führen. Dies ist nur eine der kleineren Abweichungen zwischen Dokumententypen einer Klasse. Um Schemata, die eine Klasse von Dokumententypen repräsentieren, zur Erzeugung von Formularstrukturen verwenden zu können, sind klare Bezeichnungen für Dokumententypen im Schema nötig. Für jeden Dokumententyp kann dann durch entsprechende Software eine separate Formularstruktur erstellt werden.

Innerhalb des Beispieles wurde eine weitere Schwäche der Klasse von Geschäftsdokumententypen offengelegt. Ein Fax muss eine *Faxnummer des Empfänger* enthalten, 'kann' ist nicht zulässig. Da es sich aber nicht um HTML-Formulare handelt, die die Korrektheit der Daten jedes Eingabefeldes prüfen, ist die Forderung nicht bedeutend und kann vernachlässigt werden. Eingabedaten bestehen in den analysierten Geschäftsdokument oft aus Zeichenketten, die nicht auf ihre Strukturen getestet werden können. Spezielle Datentypen sollen durch die Eingabehilfen in korrekter Form präsentiert werden. Weitere testbare spezielle Konstrukte, wie das Geschäftszeichen des Beschaffungsantrages, sind selten in Benutzung. Aus diesen Gründen wird keine Prüfung der Datentypen im System vorgesehen.

Die Formulare werden wie erwähnt Eingabehilfen beinhalten, wie die Abfrage der Adressdatenbank, das Selektieren von Mitarbeitern aus einer Liste oder die Auswahl von Alternativen oder Optionen. Für solche Typen von Elementen gibt es natürlich in keiner Spezifikation von Schemata eine Entsprechung. Daher werden sie selbst spezifiziert und signalisieren der verarbeitenden Software zur Erstellung der Formularstruktur das Auftreten der speziellen Elemente. Der Typ `address` könnte die verarbeitenden Software beispielsweise zum Auslesen der Adressdatenbank veranlassen. Die Einbindung dieser Typen und deren Elemente, die stark mit deren Verarbeitung zusammenhängt geschieht in der Implementation. Deshalb enthält der Entwurf kein fertiges Schema zur Repräsentation von XML-Geschäftsdokumenten und zur Erstellung von HTML-Formularen.

Das Problem der Erzeugung verschiedener Dokumententypen kann gelöst werden, indem das Dokumententypen-repräsentierenden Schema Zusatzinformationen über die Dokumententypen enthält. Dies kann durch zusätzliche Attribute oder eine zweckentfremdete Verwendung existierender Attribute erreicht werden. Somit wird ein Schema geschrieben, das den Konventionen der XML-Schema-Spezifikation [XML-Schema] nicht mehr ganz entspricht, aber folgende Lösung bietet.

Wie in Abbildung 5.11 dargestellt, soll ein Schema* erstellt werden, aus dem durch den Einsatz geeigneter Software für jeden Dokumententyp der Klasse eine Formularstruktur erzeugt wird. Die Art der Formularstruktur ist Teil der Implementierung. Schema* definiert eine Dokumentenklasse, die aus zwei Dokumententypen besteht.

Außerdem kann ein der XML-Schema-Spezifikation entsprechendes Schema für die Klasse oder wie in Abbildung 5.12 dargestellt für jeden Dokumententyp ein Schema erzeugt werden. In Variante zwei könnte die oben erwähnte Schwäche eines Schemas für eine Klasse von Geschäftsdokumententypen beseitigt werden.

Das Bilden von Dokumentenklassen hat letztlich einen Vorteil und einen Nachteil. Diese Lösung ist ein Kompromiss, der durch das Definieren von Schemata für jeden Dokumententyp umgangen werden kann. Trotzdem spart das Zusammenfassen ähnlicher Dokumententypen, die auch auf ähnliche Weise visualisiert werden, einen erheblichen Mehraufwand für die Definition von Schemata und Stylesheets für die Dokumententypen ein. Deshalb wird eine Software zur Verfügung gestellt, die zwei in der XML-Schema-Spezifikation enthaltenen Attribute eine andere Verwendung zukommen lässt. Attribute der XML-Schema-Spezifikation, die eine etwas andere Bedeutung und erweiterte Funktionalität erhalten, sind `name` und `fixed` und werden in der Dokumentation (Abschnitt 7.2.2) erläutert. In dieser befindet sich auch eine für die Erzeugung

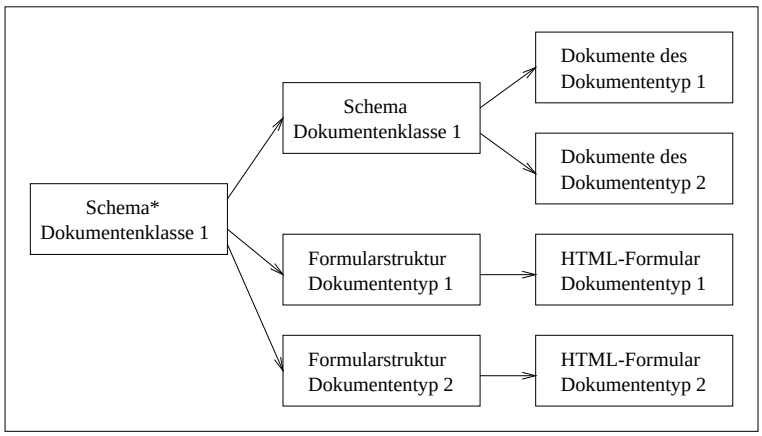


Abbildung 5.11: Zusammenhänge der Formularerzeugung 1

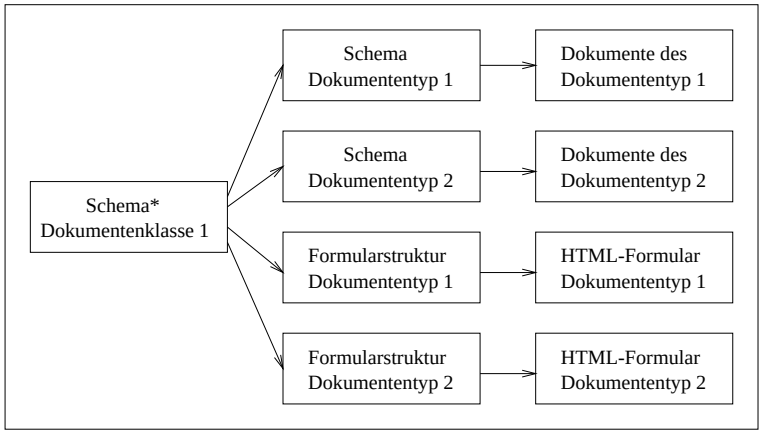


Abbildung 5.12: Zusammenhänge der Formularerzeugung 2

von Schemata* notwendige Struktur (DTD).

Wie erwähnt hängt die Definition der Schemata vor allem von der Implementation der Eingabehilfen ab. Deshalb sind sie in diesem Kapitel nicht aufgeführt. Allerdings sind die Voraussetzungen an mögliche Strukturen der Klassen von Geschäftsdokumenten bekannt. Um den Inhalt festzulegen, der durch die Analyse der Dokumententypen bestimmt wurde, werden nachfolgend die Dokumenttyp-Definitionen der gebildeten Klassen vorgestellt, die in der Implementation in verwendbare Schemata konvertiert werden. Der Inhalt steht somit fest, nur die konkrete Auszeichnung hängt unter anderem von der Implementation ab. Die Darstellung der Inhalte geschieht in Form von DTD's, da diese für den Betrachter auf den ersten Blick übersichtlicher sind.

Die Klasse Brief besteht aus den Dokumententypen Brief, Angebotseinholung und Fax.

```
<!ELEMENT Brief          (Empfänger, Betreff, Text, Gruß)>
<!ATTLIST Brief          Ihr_Zeichen             CDATA #IMPLIED
                         Ihre_Nachricht_vom      CDATA #IMPLIED
                         Unser_Zeichen           CDATA #IMPLIED
                         Unsere_Nachricht_vom    CDATA #IMPLIED
                         Bearbeiter              CDATA #IMPLIED
                         Telefon                 CDATA #REQUIRED
                         Fax                     CDATA #IMPLIED
                         Email                   CDATA #IMPLIED
```

```
                        Datum                   CDATA #REQUIRED>
<!ELEMENT Empfänger     (Adresse)+>
<!ELEMENT Adresse       (Firma, Abteilung, Vorname, Nachname, Straße,
                         Postleitzahl, Ort, Telefon, Fax, Präfix, Anrede)>
<!ELEMENT Firma         (#PCDATA)>
<!ELEMENT Abteilung     (#PCDATA)>
<!ELEMENT Vorname       (#PCDATA)>
<!ELEMENT Nachname      (#PCDATA)>
<!ELEMENT Straße        (#PCDATA)>
<!ELEMENT Postleitzahl  (#PCDATA)>
<!ELEMENT Ort           (#PCDATA)>
<!ELEMENT Telefon       (#PCDATA)>
<!ELEMENT Fax           (#PCDATA)>
<!ELEMENT Präfix        (#PCDATA)>
<!ELEMENT Anrede        (#PCDATA)>
<!ELEMENT Text          (Abschnitt, Tabelle, Liste, Artikelliste)*>
<!ELEMENT Abschnitt     (#PCDATA, Fett, Kursiv)*>
<!ELEMENT Fett          (#PCDATA)>
<!ELEMENT Kursiv        (#PCDATA)>
<!ELEMENT Tabelle       (Reihe)*>
<!ELEMENT Reihe         (Spalte)*>
<!ELEMENT Spalte        (#PCDATA)>
<!ELEMENT Liste         (Element)*>
<!ELEMENT Element       (#PCDATA)>
<!ELEMENT Artikelliste  (Artikel)*>
<!ELEMENT Artikel       (#PCDATA)>
<!ATTLIST Artikel       Stückzahl               CDATA #REQUIRED>
<!ELEMENT Gruß          (#PCDATA)>
<!ATTLIST Gruß          Unterschriftszeile_1    CDATA #REQUIRED
                        Unterschriftszeile_2    CDATA #IMPLIED>
```

Die Klasse Hausmitteilung besteht aus den Dokumententypen Hausmitteilungen und Gutachten.

```
<!ELEMENT Hausmitteilung   (Ansprechpartner, Empfänger, Betreff, Text,
                            Besteller, Stellungnahme, Sonstiges, Gruß)
<!ATTLIST Hausmitteilung   Datum                CDATA #REQUIRED
                           Mein_Zeichen         CDATA #IMPLIED>
<!ELEMENT Ansprechpartner  (Mitarbeiter)+>
<!ELEMENT Mitarbeiter      (Vorname, Nachname, Telefon, Kürzel, Email)>
<!ELEMENT Vorname          #PCDATA)>
<!ELEMENT Nachname         #PCDATA)>
<!ELEMENT Telefon          #PCDATA)>
<!ELEMENT Kürzel           #PCDATA)>
<!ELEMENT Email            #PCDATA)>
<!ELEMENT Empfänger        (Adresse)+>
<!ELEMENT Adresse          (Fakultät, Professur, Name, Anrede)>
<!ELEMENT Fakultät         #PCDATA)>
<!ELEMENT Professur        #PCDATA)>
<!ELEMENT Name             #PCDATA)>
<!ELEMENT Anrede           #PCDATA)>
<!ELEMENT Text             (Abschnitt, Tabelle, Liste)*>
<!ELEMENT Abschnitt        (#PCDATA, Fett, Kursiv)*>
<!ELEMENT Fett             (#PCDATA)>
<!ELEMENT Kursiv           (#PCDATA)>
<!ELEMENT Tabelle          (Reihe)*>
<!ELEMENT Reihe            (Spalte)*>
<!ELEMENT Spalte           (#PCDATA)>
<!ELEMENT Liste            (Element)*>
<!ELEMENT Element          (#PCDATA)>
<!ELEMENT Besteller        (vom, Fakultät, Professur, Geschäftszeichen, Preis)>
<!ELEMENT vom              (#PCDATA)>
<!ELEMENT Fakultät          (#PCDATA)>
<!ELEMENT Geschäftszeichen (#PCDATA)>
<!ELEMENT Preis            (#PCDATA)>
<!ELEMENT Stellungnahme    (Eintrag)*>
<!ELEMENT Eintrag          (#PCDATA)>
```

```
<!ELEMENT Sonstiges        (#PCDATA)>
<!ELEMENT Gruß             (#PCDATA)>
<!ATTLIST Gruß             Unterschriftszeile_1    CDATA #REQUIRED
                           Unterschriftszeile_2    CDATA #IMPLIED>
```

Die Zugangsberechtigung stellt einen Dokumententypen dar.

```
<!ELEMENT Zugangsberechtigung   (Mitarbeiterliste)
<!ATTLIST Zugangsberechtigung   Datum           CDATA #REQUIRED
                                Gültig_bis      CDATA #REQUIRED>
<!ELEMENT Mitarbeiterliste      (Mitarbeiter)+>
<!ELEMENT Mitarbeiter           (Präfix, Vorname, Nachname)>
<!ELEMENT Präfix                (#PCDATA)>
<!ELEMENT Vorname               (#PCDATA)>
<!ELEMENT Nachname              (#PCDATA)>
```

Der Beschaffungsantrag stellt einen Dokumententypen dar.

```
<!ELEMENT Beschaffungsantrag   (Artikelliste, Empfänger, Lieferfirma,
                                Bedarfsbegründung, Sonstiges)
<!ATTLIST Beschaffungsantrag   Datum              CDATA #REQUIRED
                               Bearbeiter         CDATA #REQUIRED
                               Fernsprecher       CDATA #REQUIRED
                               Geschäftszeichen   CDATA #REQUIRED
                               Inventarisierung   CDATA #REQUIRED
                               Kapitel            CDATA #REQUIRED>
<!ELEMENT Artikelliste         (Artikel)+>
<!ATTLIST Artikelliste         Mehrwertsteuer     CDATA #REQUIRED>
<!ELEMENT Artikel              (#PCDATA)>
<!ATTLIST Artikel              Menge              CDATA #REQUIRED
                               Preis_je_Einheit   CDATA #REQUIRED>
<!ATTLIST Empfänger            (Vorname, Nachname, Raum, Telefon)>
<!ELEMENT Vorname              (#PCDATA)>
<!ELEMENT Nachname             (#PCDATA)>
<!ELEMENT Raum                 (#PCDATA)>
<!ELEMENT Telefon              (#PCDATA)>
<!ELEMENT Lieferfirma          (Firma, Straße, Postleitzahl, Ort, Telefon, Fax)>
<!ELEMENT Firma                (#PCDATA)>
<!ELEMENT Straße               (#PCDATA)>
<!ELEMENT Postleitzahl         (#PCDATA)>
<!ELEMENT Ort                  (#PCDATA)>
<!ELEMENT Telefon              (#PCDATA)>
<!ELEMENT Fax                  (#PCDATA)>
<!ELEMENT Bedarfsbegründung    (#PCDATA)>
<!ELEMENT Sonstiges            (#PCDATA)>
```

5.5 Zusammenfassung

Das eOffice-System wird auf der Basis des CGI-Konzepts unter Verwendung der Skriptsprache Tcl erstellt. Ausschlaggebend für diese Wahl waren unter anderen die für die interne Verarbeitung benötigten Pakete TclXML und TclDOM, die eine Erweiterung von Tcl darstellen. Das System stellt eine auf Sessions basierende Intranet-Anwendung zur Erledigung von Geschäftsvorfällen dar, die eine Web-Nutzerschnittstelle bietet. Der Hauptgedanke des Entwurfes ist die dynamische Erzeugung von HTML-Formularen aus Formularstrukturen. Diese werden vom System aus Schemata generiert, die einer vereinfachte Version der XML-Schema-Spezifikation entsprechen. Diese Schemata beschreiben die Struktur und den Inhalt der Geschäftsdokumente. Auf das Erstellen von Schemata zur Repräsentation von Dokumentenklassen wird aus zwei Gründen verzichtet.

1. Vorhandene XML-Dokumente können mit dem XML-Parser TclXML nicht gegen externe DTD's oder Schemata auf Typenkorrektheit validiert werden. (Abschnitt 6.1.2) Außerdem bestehen die Eingabedaten überwiegend aus nicht testbarem Text oder durch Eingabehilfen korrekt vorgegebener Daten.

2. Die dynamisch generierten HTML-Formulare und die Implementation bürgen für eine korrekte Erstellung der XML-Dokumente, sodass eine Überprüfung der XML-Strukturen nicht nötig ist.

Eingabedaten aus HTML-Formularen werden mittels dem Document Object Model in XML-Strukturen eingebunden, sodass erzeugte Geschäftsdokumente als XML-Datei vorliegen. Diese werden mittels einer XSLT-Engine und Stylesheets in HTML zur Voransicht und PostScript als druckfertige Version konvertiert.

Eine Erweiterung des Systems um neue Bürodokumente kann einfach durch die Definition von Schemata und Stylesheets geschehen. Zur Unterstützung der Nutzer beim Ausfüllen der Formulare sind Eingabehilfen und die Anbindung an eine Adressdatenbank geplant.

Kapitel 6

Implementierung

6.1 Programmieren mit Tcl

Die Tool Command Language (Tcl) ist eine robuste und weit verbreitete Skriptsprache, die in letzter Zeit auch stark für CGI-Programmierung eingesetzt wird. Sie ist eine interpretierende Sprache und sehr beliebt, weil sie sich durch eine einfache Syntax auszeichnet, frei für kommerzielle und nicht-kommerzielle Zwecke verwendet werden kann und es eine Reihe von meist auch freien nützlichen Erweiterungen gibt, die die Arbeit mit Tcl erleichtern. Tcl kann für jede Art der Erstellung von Web-Applikationen verwendet werden, sowohl auf der Klienten- als auch auf der Serverseite und bietet das Paket TK zur Erstellung grafischer Nutzeroberflächen. Informationen über und neueste Versionen von Tcl stellt die Seite `www.scriptics.com` bereit. Über Kommandos und Erweiterungen gibt „Tcl/Tk in a Nutshell“ [Rai99] einen schnellen Überblick. Um die in der weiteren Arbeit angeführten Beispiele ohne Tcl-Kenntnis zu verstehen, sind einige Beispielzeilen nachfolgend angeführt. Alle Variablendaten werden in Tcl grundsätzlich als Strings verstanden. Mit `set` werden einfachen Variablen, Listen und Feldern Werte zugewiesen, die beispielsweise durch das Ausführen einer Funktion geliefert werden.

```
set var_a wert1
set var_b wert2
set list {$var_a $var_b}
set res(compare) [compare $list]
```

Sollten die obigen Definitionen in einem separaten File spezifiert sein, kann das Hauptprogramm durch ein Laden der Datei mit `source` über die Variablen verfügen.

6.1.1 CGI-Programmierung

Tcl-CGI Skripts verwenden die `tclsh` und liegen ausführbar mit einer definierten Namenserweiterung (meist .cgi) in einem für CGI's konfigurierten Verzeichnis eines Web-Servers. Für die Programmierung von CGI's gibt es einige Erweiterungen. Die Implementation verwendet das sehr umfangreiche Paket `cgi.tcl` von Don Libes, welches die CGI-Programmierung erleichtert. Nähere Informationen sind unter [Lib96] zu finden und in „WebTcl Complete“ [Bal99] beschrieben. Als kurze Beispiele sollen die Erzeugung einer Liste und eines Links genügen. Die Tcl-Anweisungen

```
foreach elem {Eins Zwei Drei} {
  cgi_number_list {
    cgi_li $elem
  }
}
```

erzeugen:

```
<ol>
  <li>Eins</li>
  <li>Zwei</li>
  <li>Drei</li>
</ol>
```

Einen Bilderlink kann man auf folgende Weise definieren:

```
cgi_link muster [cgi_imglink bild Bild.jpg "alt=Text"] Datei.html
```

Der Einsatz des definierten Links `muster` durch

```
put "[link muster]"
```

erzeugt die Zeile

```
<a href="Datei.html"><img src="Bild.jpg" alt="Text"></a>
```

im zurückzusendenden Dokument.

Ist der Link in einer separaten Datei gespeichert, kann er durch das Einlesen dieser Datei in beliebige Tcl-Skripts verwendet werden. Falls der Link mehrfach benutzt wird, lässt sich die Ersparnis an Änderungen schon an diesem kleinen Beispiel erkennen.

6.1.2 Verwendung von TclDOM/TclXML

Zur internen Verarbeitung von Eingabedaten wird das Paket TclDOM verwendet. Es ist eine Document Object Model-Anbindung an Tcl, die XML-Dateien in einer Baumstruktur repräsentieren kann. Die verwendete Version von TclDOM unterstützt nur XML-Dokumente und implementiert den DOM Level 1, der genügend Methoden zur Manipulation von XML-Datenstrukturen bietet. TclDOM erzeugt einen DOM-Baum, führt Aktionen auf diesem aus und kann XML-formatierten Text ausgeben.

Im folgenden Beispiel wird ein neues DOM-Objekt erzeugt und serialisiert. `doc`, `top` und `node` sind Token, die gesetzt werden, um sie im weiteren Aufbau des Dokumentes zu referenzieren.

```
set doc  [dom::DOMImplementation create]               # neues Objekt
set top  [dom::document createElement $doc Root]       # Wurzel-Element
set node [dom::document createElement $top Erstes]     # neues Element
dom::document createTextNode $node "Inhalt"            # Textelement
dom::DOMImplementation serialize $doc                  # Serialisierung
```

Diese Anweisungen erzeugen folgendes XML-Dokument:

```
<?xml version="1.0"?>
<!DOCTYPE Root>
<Root>
  <Erstes>
    Inhalt
  </Erstes>
</Root>
```

Die DOM-Implementation TclDOM erzeugt eine hierarchische Struktur, einen Baum. Um das dabei entstandene Eltern-Kind-Verhältnis zu verdeutlichen, wird die erzeugte Struktur in Verbindung mit einem XML-Dokument im folgenden Beispiel aufgezeigt.

```
<?xml version="1.0"?>        xml                    node1
  <DOCTYPE Root>               doctype              node2
  <Root>                       element (Root)       node3
                               text                 node4
    <Erstes>                     element (Erstes)   node5
      Inhalt                       text (Inhalt)    node6
    </Erstes>
                               text                 node7
  </Root>
```

Wie dargestellt, werden sieben Nodes erzeugt, die per Token als Identifikatoren für Knoten vom Programmierer angesprochen werden können. Ein Token hat die Form `::document4(node3)`. Auf den Inhalt sollte aber nicht direkt über ein Token, sondern über entsprechend zur Verfügung stehende Methoden zugegriffen werden. Das Wurzel-Element besitzt im Beispiel drei Kinder (`node4,node5,node7`). Das zweite selbst eingefügte Kind `<Erstes>` ist von den Textknoten `node4` und `node7` umgeben, die von der DOM-Implementation automatisch angelegt werden und DOM1-konform sind. Neben den in DOM Level 1 empfohlenen Zugriffsmethoden sind in TclDOM weitere Methoden implementiert, die die Arbeit mit XML-Dokumenten erleichtern. [TclDOM]

Zum Parsen der XML-Dokumente wird das Paket TclXML verwendet, das wie TclDOM auch vollständig in Tcl implementiert ist. Es hat allerdings in der Version 1.2 den Nachteil, dass das Parsen von XML DTD's und somit von XML Parameter-Entities nicht unterstützt wird. [TclXML] (Die fünf allgemeinen Entities `<` (<), `>` (>), `"` (\), `'` (') und `&` (&) werden automatisch substituiert.) Falls Parameter-Entities im Dokument vorkommen, bleiben sie gemäß der XML-Spezifikation unberührt und werden ignoriert. TclXML ist ein somit ein XML-Parser, der XML-Dokumente nicht gegen eine externe DTD oder ein Schema validieren kann. Wie im weiteren Verlauf der Arbeit aber erklärt wird, ist der XML-Parser TclXML dennoch für den Gebrauch im eOffice ausreichend.

TclDOM reagiert jedoch auf alle Entities in einer für die weitere Bearbeitung ungeeigneten Form. Es konstruiert beim Parsen eines XML-Dokumentes mit beispielsweise dem Element

```
<Abschnitt>foo &amp; bar</Abschnitt>
```

nicht einen Textknoten für das Abschnitt-Tag, sondern zerlegt den Inhalt in die drei Textknoten `foo`, `&` und `bar`. Dies geschieht ebenso für substituierten wie auch für nicht substituierte Entities. Die weitere Verarbeitung dieser entstandenen Knoten gestaltet sich schwierig, wenn der Gesamttext zum Editieren selektiert werden soll. (Die Techniken der Verarbeitung werden später im Kapitel erklärt.) Aus diesem Grund wird in der internen Verarbeitung auf das Ampersand-Zeichen & verzichtet, dass die Entities als solche kennzeichnet und dadurch die Zerlegung in mehrere Textknoten hervorruft. Der Verzicht auf das Zeichen erfordert eine Zeichenersetzung. (Abschnitt 6.4.5)

TclXML stellt Parser-Dienste zur Verfügung und wird in den Paketen `xml.tcl` und `sgml.tcl` implementiert. Das Document Object Model wird in `dom.tcl` mit seinen Zugriffsmethoden an die Skriptsprache Tcl gebunden.

6.2 Philosophie des Systems

Wie schon erwähnt, ist es nur schwer möglich, Geschäftsdokumente von der Papierform in gleichaussehenden elektronischen Formularen abzubilden, die dem Nutzer als Web-Interface zur Eingabe zur Verfügung stehen sollen. HTML-Formulare des eOffice werden dynamisch für jeden CGI-Request aus definierten Formularstrukturen erstellt. Die Formulargenerierung und die Einbindung von Daten aus einem HTML-Formular in XML-Struktur sind Eckpunkte der Philosophie des Systems.

6.2.1 Grundlage

Die Grundlage des Systems führt auf Überlegungen zur Implementation eines elektronischen Büros[1] von Herrn Prof. Hübner zurück, deren Grundgedanken sich in zwei Punkten darstellen lassen.

[1] Ansatz zur „elektronischen Geschäftsabwicklung“: http://www-usercgi.tu-chemnitz.de/~huebner/cgi-bin/eoffice.cgi

1. Geschäftsdokumente lassen sich gliedern und zur Repräsentierung in XML hierarchisch aufbauen. Diese Gliederung kann in eine Variablenstruktur überführt werden, welche der Generierung von HTML-Formulare dient. Durch die Hierarchie kann mittels Formular-Buttons navigiert werden. Der aktuelle Standpunkt wird in einem Pfad angegeben.

2. XML-Strukturen bestehen aus Elementen und Attributen. Die Eingabe von Textdaten in HTML-Formulare geschieht durch mehrzeilige Eingabefelder (Textarea) und einzeilige Eingabefelder (Input). Um die Daten aus Formularen in geeigneter Weise in XML-Strukturen zu überführen, kann ein XML-Element den Inhalt des Textareas erhalten und die Attribute eines Elementes können die Einträge aus den Input-Feldern aufnehmen. Für ein Element in XML wird ein HTML-Formular generiert.

Dieser Ansatz gab die Richtung zur Implementation vor und hat einige Vorteile gegenüber anderen Vorgehensweisen. Die Darstellung eines gesamten statischen Formulars für ein Geschäftsdokument ist nicht sinnvoll, da jeder Brief vor allem im Textbereich anderes aussieht. Dies kann ein statisches Formular nicht abdecken. Es ist allerdings für die Erstellung echter Papierformulare anwendbar, wenn alle Elemente fest vorgeschrieben sind. Eine weitere Variante ist ein Template-Dokument als Mischung zwischen statischen Formular und dynamischer Elementenerzeugung. Dieses Template definiert auf Grund einer XML-Struktur statische Eingabefelder, kann aber auch vom Nutzer definierte Eingabefelder, z.B. Tabellen, erhalten. Auch diese Variante ist unübersichtlich, da das Formular auf eine Web-Seite passen muss. Konstruktiv weitergedacht führt das Template wiederum zum oben vorgestellten Ansatz.

6.2.2 Erweiterung

Die Bearbeitung eines Geschäftsdokumentes geschieht im eOffice durch Navigieren in der hierarchischen Datenstruktur des Dokumentes. Somit müssen für ein Dokument mehrere verschiedene HTML-Formulare generiert werden. Um den Zusammenhang zu XML herzustellen, erfolgt das Erzeugen eines Formulars für jedes XML-Element. Die Attribute dieses Elementes werden im HTML-Formular als einzeilige Eingabefelder berücksichtigt. Für die interne Präsentation des Dokuments wurden Konstrukte definiert, die die Zusammensetzung der Dokumente und somit der einzelnen Formulare verdeutlichen und die das Tcl-Skript verarbeiten kann. Am Beispiel der DTD des Briefes (Abschnitt 5.4) werden Tcl-Felder für Elemente und Attribute definiert, die der Dokumenttyp-Definition stark ähneln.

```
set elemente(Brief)     {Empfänger Betreff Text Gruß}
set attribute(Brief)    {Ihr_Zeichen Ihre_Nachricht_vom ... Email Datum}
set elemente(Empfänger) {Adresse}
set elemente(Adresse)   {Firma Abteilung Vorname Nachname ... }
```

Wie schon erwähnt gibt es weitere Anforderungen an die auftretenden Strukturen. Zum einen werden neue Typen für Eingabehilfen definiert, die nicht als Texteingabefeld im HTML-Formular repräsentiert werden können und zum anderen besteht die Möglichkeit, dass Elemente mehrfach auftreten können (z.B. Adressen für Serienbriefe) Beide Merkmale müssen dem verarbeitenden Programm zur Einbindung und Manipulation von XML-Daten mitgeteilt werden. Da nur die Elemente mehrfach vorkommen und die Formularerzeugung von ihnen abhängt, wird festgelegt, dass nur die Elemente durch einen weiteren Typ und einer Kennzeichnung über ihr Auftreten (einfach/mehrfach) näher bestimmt werden. Neue komplexe Typen wurden im Entwurf erwähnt und sind auf Grund der Analyse der Geschäftsdokumente notwendig. Außer dem einfachen Text (string) sind vorgesehen:

- Adressenauswahl aus Listen (address)
- Mitarbeiterauswahl aus Listen (mitarbeiter)
- Struktureinheitenauswahl aus Fakultät und Professur (facproflist)

- Auswahl von Alternativen (radiobutton)
- Auswahl von Optionen (checkbox)

Für das Auftretens eines Elementes gibt es die zwei Möglichkeiten „einfach“ oder „mehrfach“. Um dies für die interne Verarbeitung zu kennzeichnen, wird „einfach“ auf „+“ und „mehrfach“ auf „*“ gesetzt. Zur Repräsentation der beiden Merkmale wird für alle Elemente ein weiteres Tcl-Feld eingeführt. Es enthält Typ und Auftreten als zwei Zeichen entsprechend allen Elementen des `elemente`-Feldes. (siehe folgendes Beispiel) Deshalb beginnen die Namen der neuen Typen mit unterschiedlichen Buchstaben. DTD's sind für diese Erweiterung nicht mehr ausreichend, da unterschiedliche Typen spezifiziert werden müssen. Die Definition kann in einem Schema durch das Attribut `type` geschehen.

```
set elemente(Brief)            {Empfänger Betreff Text Gruß}
set elemente_typen(Brief)      {+a +s +n +s}
set attribute(Brief)           {Ihr_Zeichen Ihre_Nachricht_vom ... Email Datum}
set elemente(Empfänger)        {Adresse}
set elemente_typen(Empfänger)  {*n}
```

Für die interne Verarbeitung werden die im Beispiel aufgeführten Konstrukte aus den Schemata erstellt. Das neue Tcl-Feld wurde dem Einfügen der Typinformationen in die Namen der Elemente (des Feldes `elemente`) vorgezogen, weil es intern einfacher zu verwalten ist.

Um die mehrfach vorkommenden Elemente in der internen Verarbeitung unterscheiden zu können, werden sie mit Identifikatoren versehen. Dabei wurden zwei Alternativen betrachtet. Einerseits können nach XML-Schema definierte ID-Typen verwendet werden, welche eindeutig während der Verarbeitung erzeugt werden. Aus dem Pfad oder anderen Ressourcen lassen sich allerdings keine geeigneten Id's erzeugen. Anfangsbuchstaben der im Pfad enthaltenen Elemente und einer fortlaufenden Zahl ergeben beispielsweise keinen eindeutigen String. Eine verwendete zweite Variante sieht einfach die Nummerierung der Elemente vor, ist demnach kein ID-Typ im Sinne der XML-Schema Spezifikation. Dieses mit POSID bezeichnete Attribut hat aber den Vorteil, dass deren Inhalt auch im Nutzerinterface zusammen mit den Pfadelementen Auskunft über die Anzahl der erzeugten Elemente geben kann. Für den Systembetreuer entfällt die Aufgabe, ID-Attribute in den Schemata zu definieren. Dies geschieht systemintern und stellt eine Vereinfachung für den Administrator dar.

Um dem Nutzer und dem verarbeitenden Programm die aktuelle Position in der Dokumentenstruktur zu vermitteln, wird ein Pfad (Tcl-Liste) aus Elementen angelegt. Diese Liste beginnt mit dem Wurzelelement des bearbeiteten Dokumententyps und enthält weiterhin ein Element aus jeder besuchten Hierarchieebene. Dadurch wird ein Pfad im DOM-Baum gebildet. Da der einfache Pfad wieder keinen mehrfach vorkommende Elemente unterscheiden kann, wird ein weiterer Pfad für die Positions-Id's angelegt. Daraus kann das System erkennen, um welches Element es sich genau handelt. Das folgende Beispiel verdeutlicht die Theorie an einem Serienbrief, in dem der Name der dritten Adresse bearbeitet wird.

```
path      :  {Brief Empfänger Adresse Name}
posidpath :  {1     1         3       1   }
```

Da das Wurzelelement konstant die Position 1 besitzt, wird diese in der Implementation nicht berücksichtigt.

Um die geschilderten Implementation zu visualisieren, ist die Präsentation der Geschäftsdokumente von der Analyse bis zum fertigen XML-Dokument in Abbildung 6.1 dargestellt.

Das aus dem Schema erstellte HTML-Formular und die Implementation bürgen für ein korrektes Erstellen der XML-Dokumente. Deshalb ist eine DTD oder ein Schema zur Validierung nicht zwingend nötig und werden für dieses System nicht erstellt, zumal TclXML XML-Dokumente nicht gegen ein Schema validiert.

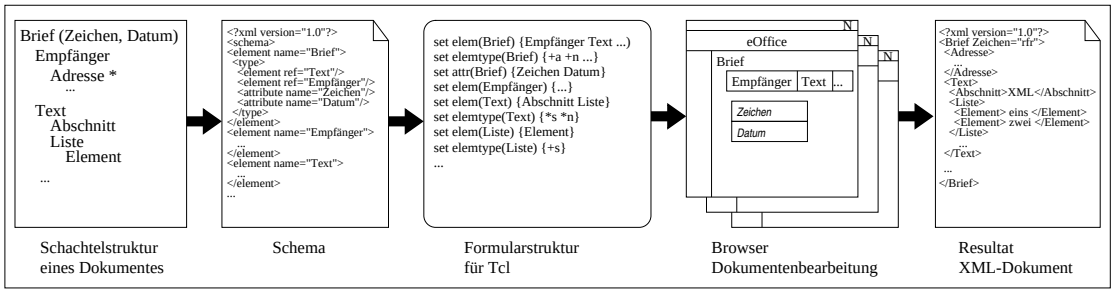


Abbildung 6.1: Philosophie des Systems

Vorteile der Philosophie

- Dynamischer Aufbau von Formularen - offen für neue Dokumente.
- Direkter Bezug zwischen HTML-Formularen und XML-Strukturen herstellbar.
- Erfassen großer auch längerer Dokumente, Eingabeformulare bleiben im Rahmen der Bildschirmgröße.
- Offen für beliebige Textgestaltung im Bereich der definierten Textgestaltungsvorgaben.

Nachteil der Philosophie

Da nur für Elemente besondere Typen vorgesehen sind, kann dies zu einer unsaubere Erstellung von XML-Dokumenten führen. Das bedeutet, dass die Konvention zur Erstellung von Elementen und Attributen nicht korrekt eingehalten wird. Attribute verstehen sich als Eigenschaften und Elemente als Teile der hierarchisch übergeordneten Elemente. Die Daten einer Auswahl von Alternativen (Radiobutton) werden aber beispielsweise in einem eigenen Formular dargestellt. Somit wird intern auch ein neues Element für die selektierte Alternative erstellt und kein Attribute, welches möglicherweise der XML-Konvention für Elemente und Attribute entsprochen hätte. Deshalb ergeben sich zu den in Abschnitt 5.4 aufgeführten DTD's noch Änderung bei der Konvertierung in für das System nutzbare Schemata.

6.3 Bestandteile des Systems

Als Basis der Verarbeitung von Daten ist die CGI-Technologie gewählt worden. Das System besteht aus den Programmen `eoffice.cgi, postscript.cgi, html.cgi` und `mail.cgi`. Diese greifen auf Funktionen und Variablen aus `proc.tcl` und `header.tcl` zurück. Weiterhin werden JavaScript-Funktionen verwendet, die in `eoffice.js` enthalten sind. Die Struktur des entwickelten Systems ist vereinfacht in Abbildung 6.2 zu sehen.

Beim Aufruf von `eoffice.cgi` können je nach dynamisch erzeugtem Formular die Übergabeparameter `xml` (Serialisertes XML-Dokuemnt), `sub` (Submit-Variable für Image-Buttons), `subb` (Submit der Formular-Submit-Buttons), `fileprefix` (Nutzerspezifische Dateibezeichnung), `path` (Pfad im DOM-Baum), `posidpath` (Positionen der Elemente entsprechend des Pfades), `session` (Session-Id), `node` (Knoten im DOM-Baum), `edit` (Nutzer editiert), `print` (Anzeige der Änderungsansicht), `text` (Daten des Textareas), `attr_*` (Input-Felder), `addresslist` (Adresseneinträge), `mitarbeiterlist` (Mitarbeitereinträge) und `checkboxlist` (Checkboxeneinträge) zum CGI gesendet werden. An das generierte HTML-Formular übergibt `eoffice.cgi` davon in aktualisiertem Zustand `xml, path, posidpath` und reicht `session, edit, print, node` durch. Für die Erzeugung eines PostScript-Dokumentes benötigt `postscript.cgi` die Parameter `xml, session` und `path`. `html.cgi` erhält zur Generierung eines HTML-Previews die Werte der selben Variablen. Zum Absenden einer Mail ist für `mail.cgi session, path` und `mailLink` nötig.

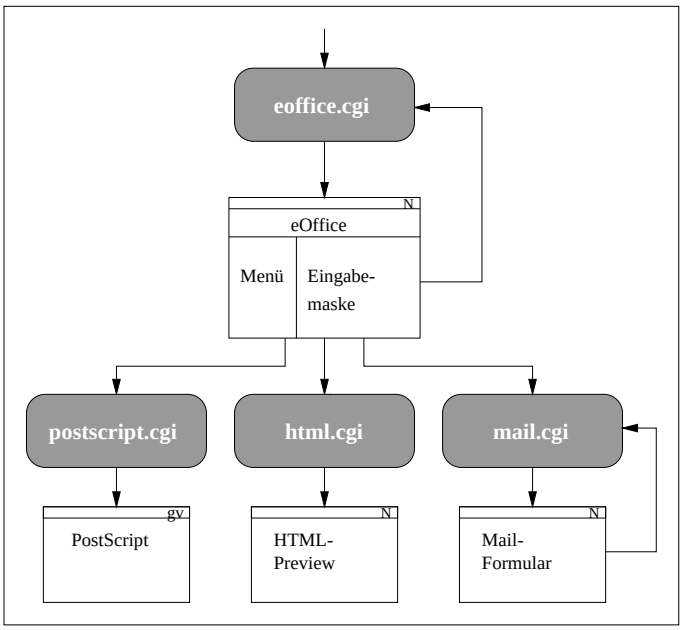


Abbildung 6.2: Struktur des Prototyps

Die CGI-Skripte implementieren folgende Aspekte:

- `eoffice.cgi`
  - Session-Management
  - Hauptmenü-Funktionalität
  - Administratortätigkeiten
  - Bearbeitung und Änderung von Geschäftsdokumenten in HTML-Formularen
  - Dokumenten-Funktionalitäten
  - Interne Verarbeitung von Formulardaten und Erzeugen von XML-Dokumenten
- `postscript.cgi`
  - Konvertierung von XML-Dokumenten in PostScript-Dokumente
- `html.cgi`
  - Konvertierung von XML-Dokumenten in HTML-Dokumente
- `mail.cgi`
  - Versenden einer Mail mit einem Link eines durch das eOffice erstellten Geschäftsdokumentes.

In den folgenden Abschnitten wird die Implementation elementarer Funktionalitäten der genannten Themengebiete erläutert.

6.4 Beschreibung elementarer Funktionalitäten

6.4.1 Voraussetzungen

Session-Konzept und Lock-Files

Die Erledigung von Geschäftsvorfällen geschieht im eOffice in Sitzungen. Für jede Session wird ein eindeutiger Sitzungsidentifikator erzeugt, der einem Ausschnitt des Fingerprints einer Zeichenkette aus `gpg`-erzeugten Zufallswerten entspricht. Für diese Anwendung sollte der Id ausreichend sein, auch wenn der durch MD5 erzeugte Hashwert nicht kollisionsresistent ist. Da die Authentifikation am System per .htaccess-Mechanismus erfolgt, wird eine Session in zwei möglichen Fällen gestartet. Zum einen geschieht dies direkt nach der Authentifikation und zum anderen, wenn der eOffice-URL ohne Parameter wiederholt vom authentifizierten Nutzer gewählt wird. Da sich der Nutzer im zweiten Fall bereits authentifiziert hatte, ist sein Nutzerkennzeichen aus REMOTE_USER zu erfahren. Eine erneute Authentifikation soll dem Nutzer erspart bleiben, da ein Missbrauch des Systems bei Abwesenheit des Nutzers nicht von sicherheitskritischer Relevanz ist. Mit der Erzeugung einer neuen Session wird ein Session-File angelegt, in dem das Nutzerkennzeichen, die Rechte zum Erstellen von Geschäftsdokumenten und das Administratorrecht als Tcl-Code definiert werden. Die Rechte werden dem Mitarbeiterverzeichnis entnommen. Das Session-File besitzt den Namen der Session-Id und erhält die Erweiterung `.tcl` und kann per `source` in Tcl-Skripte eingelesen werden. Alle Dateien, die Sitzungen repräsentieren, werden in einem separaten Verzeichnis gespeichert. In der aktuellen Implementation ist das Nutzerkennzeichen durch die Umgebungsvariable immer präsent. Bei einem Einsatz eines anderen Authentifikations-Mechanismus, der die Umgebungsvariable REMOTE_USER nicht setzt, ist das Nutzerkennzeichen nur noch durch das Session-File zu erfahren. Der Sitzungsidentifikator wird im HTML-Formular exportiert, d.h. durch eine versteckte Variable weitergegeben. Im ausführenden CGI-Programm wird die Session-Id überprüft und das Session-File per `source` geladen, womit die in Tcl definierten Variablen verwendbar sind. Diese Session-Dateien werden ab einer definierten Zeit gelöscht. Der Default-Wert beträgt 10 Stunden und entspricht etwa der täglichen Arbeitszeit. Die alten Session-Dateien werden nur gelöscht, wenn ein neuer Session-Id erzeugt wird, da diese Funktion auch über die Existenz der Session-Files befindet. Falls einem authentifizierten Nutzer die Sitzungsdatei gelöscht wird, fällt dies beim nächsten Request auf und er erhält eine neue Session. Bei den derzeit im Session-File gespeicherten Informationen zeigt sich dem Nutzer kein Unterschied. Definierte Zustände in der Datei würden auf ihre Default-Werte zurückgesetzt.

Wenn ein Nutzer ein Dokument öffnen möchte, welches gerade in Bearbeitung ist, dann wird ihm das durch einen ähnlichen Mechanismus wie bei `vi` mitgeteilt. Das Dokument kann durch einen anderen Mitarbeiter oder durch den Nutzer selbst erneut bearbeitet werden. Beim Öffnen und Speichern als neuen Namen wird zur bearbeiteten Datei ein Lock-File angelegt, das den gleichen Stammnamen aber eine andere Extension besitzt. Diese ist aus *„Nutzerkennzeichen“.„Session-Id“.lock* aufgebaut und damit für jede Session eindeutig. Das enthaltene Nutzerkennzeichen wird beim Öffnen zur Ansicht gebracht, um den neuen Nutzer auf die Personen hinzuweisen, die dieses Dokument ebenso bearbeiten. Falls das Dokument nicht geschlossen wird, entweder weil während der Bearbeitung ein anderer URL angewählt wurde oder der Browser abgestürzt ist, bleibt das Lock-File für eine Mindestzeitdauer erhalten. Alte Lock-Dateien werden beim nächsten Versuch, dieses Dokument zu öffnen, durch eine eingebaute Routine gelöscht. Ebenso geschieht dies durch das Schließen von Dokumenten. Beim Speichern als neuen Dateinamen wird die vorher bearbeitete Datei geschlossen.

Nutzerinterface

Das Nutzerinterface ist aus HTML-Tabellen aufgebaut, wobei die Gesamttabelle aus zwei Zeilen besteht und die zweite Zeile in zwei Spalten aufgeteilt ist. (siehe Abbildung 5.6) Die Zellen enthalten wiederum weitere Tabellen.

JavaScript für Menü und Eingabehilfen

Durch den Einsatz von JavaScript können grafische Buttons verwendet werden, die das eOffice optisch attraktiver erscheinen lassen. `eoffice.js` beinhaltet JavaScript-Funktionen, die die Navigation durch Image-Buttons ermöglichen und eine Eingabehilfe zur einfachen Auswahl einer Struktureinheit implementiert. Alle Parameter, die durch JavaScript-Funktionen geändert werden sollen, müssen im HTML-Formular übergeben werden, d.h. entweder als „hidden"-Variablen oder als Eingabemöglichkeit zur Verfügung stehen. Andernfalls ist eine Wertzuweisung nicht möglich, da die Parameter für JavaScript keine änderbaren Eigenschaften besitzen.

6.4.2 Funktionen des Hauptmenüs

Im Hauptmenü gibt es die vier Funktionen Neu, Öffnen, Suchen und Administration. (siehe Abbildung 5.8) Ein fünfter Button Hilfe öffnet eine HTML-Hilfeseite. Der Button Administration erscheint nur, wenn der authentifizierte Nutzer das Recht zur Administration des Systems besitzt. Durch das Betätigen dieses gelangt der Nutzer in ein neues Menü, das der Administration. Aufgaben des Systembetreuers (z.B. neue Dokumenttypen zur Verfügung stellen oder das eOffice-interne Mitarbeiterverzeichnis aktualisieren) sind Thema des Abschnitts 6.4.3

Neu: Wenn sich der Nutzer für das Erzeugen eines Dokumentes entscheidet, muss er in der Eingabemaske „Neu" (Abbildung 5.8/(a)Neu) einen Button mit dem Namen des gewünschten Dokumentes betätigen. Die Dokumententypen werden einer Datei (`root.tcl`) entnommen, die durch die Verarbeitung eines neuen Schemas erzeugt wird. Buttons für Dokumententypen werden nur angezeigt, wenn der Nutzer die Zugriffsrechte auf die jeweiligen Typen besitzt. Die Rechte sind im Mitarbeiterverzeichnis definiert und werden für eine Session gesetzt. Nach der Auswahl eines Dokumentes erzeugt das System ein neues DOM-Objekt, welches als Root-Element den selektierten Namen erhält. Vorerst trägt das Programm einen automatisch generierten Dateiname als das Attribut `ID` des Root-Elementes ein. Zur Erzeugung eines neuen Dokumentes wird in das Arbeitsmenü gewechselt und ein Formular auf Grund der für das Dokument erzeugte Formularstruktur generiert, welche zuvor geladen wurde.

Öffnen: Das System unterstützt das Öffnen und die anschließende Weiterbearbeitung gespeicherter Dokumente. Der Nutzer selektiert dazu in der Eingabemaske Öffnen (Abbildung 5.8/(a)-Öffnen) einen Dateinamen in einem Auswahlfeld, in welchem er sich alle Dokumente eines Dokumententyps (abhängig von Erstellungsjahr und Nutzerkennzeichen) anzeigen lassen kann, sofern er die Berechtigung für diesen Typ besitzt. Beim Öffnen wird ein vollständiger Dateiname (incl. Verzeichnis) an das System übergeben. Diese Datei wird geparst und ein DOM-Baum erstellt. Die Formularstruktur für den Dokumententyp, deren Name sich aus den übergebenen Daten ergibt, wird geladen, und ermittelte Dokumentendaten im einem HTML-Formular dargestellt. Für das geöffnete File wird eine Lock-Datei angelegt.

Suchen: Das Suchen gestattet dem Nutzer, sich Dokumente eines Dokumententyps zusätzlich den Kriterien des Öffnen nach im Dokument vorhandenen Text anzeigen zu lassen. Dazu wurden die Systembefehle `grep` und `find` verwendet. Die in einem Auswahlfeld aufgeführten Dateien können wiederum geöffnet werden. (siehe Öffnen)

Die Nutzerrechte sind im Mitarbeiterverzeichnis definiert und werden beim Anlegen einer neuen Sitzung für den authentifizierten Nutzer ausgelesen und im Session-File abgelegt. Die Zugriffsrechte für die Dokumententypen sind durch das Einlesen dieser Datei in jedem CGI-Ablauf immer präsent. Sie werden in der Diplomarbeit nach Absprache mit den Betreuern nicht definiert. Dies ist Aufgabe eines Systembetreuers.

6.4.3 Tätigkeiten des Administrators

Generierung der Formularstruktur

Die Formularstruktur dient dem CGI-Skript dazu, abhängig von einem Pfad im DOM-Baum das entsprechende HTML-Formular dynamisch zu generieren. Der Aufbau der Struktur wurde

in Abschnitt 6.2.2 erläutert. Während des Parsens eines Schemas, welches die Struktur und Inhaltstypen eines Geschäftsdokumentes repräsentiert, wird intern ein DOM-Baum des Schemas aufgebaut. Dieser wird rekursiv durchlaufen und durch Auswertung der Elemente und Attribute die angesprochene Formularstruktur erstellt.

Die Dateien für das Schema und die Stylesheets für die Umwandlung nach LaTeX und HTML müssen auf den eOffice-Server geladen (Upload-Funktionalität) werden, um eine Verarbeitung des Schemas zu erlauben. Dazu steht ein entsprechendes Interface (Abbildung5.9/(a)Schema) zur Verfügung. Die Styles werden nur kopiert und nicht auf Korrektheit überprüft. Bei Fehlern in der Abarbeitung des Schemas werden entsprechende Meldungen angezeigt, die dem Administrator die Suche der Fehlerquelle erleichtern. Neu generierte Formularstrukturen werden als `'Dokumententyp'.tcl` in einem Verzeichnis für Formulardaten abgelegt. Diese Datei beinhaltet die für die Erzeugung von HTML-Formularen notwendigen Felder für Elemente und Attribute und die Klasse des Dokumententyps. Das File `root.tcl` enthält alle Dokumententypen, die sortiert eingegliedert werden.

Natürlich kann der Administrator die Formularstruktur auch von Hand anlegen und `root.tcl` ändern, um ein neues Dokument in das eOffice zu integrieren. Durch den Entwurf eines Schemas bestehen allerdings die Aussichten, Geschäftsdokumente gegen dieses zu validieren (falls dies von einem XML-Parser unterstützt wird). Das Schema erleichtert dem Systembetreuer somit die Arbeit, da daraus ein W3C-konformes Schema und die Formularstruktur gebildet werden könnten. Die Erstellung von Schemata und Stylesheets ist in der Dokumentation (Abschnitt 7.2.2) ausgeführt. Schemata und XSL-Styles für die analysierten Geschäftsdokumenten befinden sich auf der zur Diplomarbeit beigefügten CD.

Wie schon erwähnt, entsprechen die für die Verarbeitung im eOffice verwendeten Schemata nicht der XML-Schema Spezifikation [XML-Schema]. Die Schemata*, wie sie in Abschnitt 5.4 bezeichnet wurden, basieren auf XML-Schema, enthalten aber Unterschiede in der Bedeutung und Verwendung von Tags. Zum einen erleichtert dies die Arbeit des Systembetreuers bei der Definition von Schemata und zum anderen vereinfacht das Schema* die interne Verarbeitung erheblich. Die benötigten Typen zur HTML-Darstellung (z.B. Default-Einträge für Checkbox-Elemente) lassen sich in XML-Schema schlecht definieren, was auch nicht der Zweck der Schema-Spezifikation ist. Für Schema* wurden einfache von XML-Schema abgeleitete Konventionen gefunden, die dies realisieren. Die Verwendung der benötigten Tags zur Definition der eOffice-Schemata ist in Abschnitt 7.2.2 dargelegt. Dieser Themenbereich stellt einen Ansatzpunkt für weitergehende Überlegungen dar.

Administrative Mitarbeiterverwaltung

Der Grund für das Anlegen eines eOffice-internen Mitarbeiterverzeichnisses liegt in der Notwendigkeit der Definition von Rechten für den Dokumentenzugriff und die Administratortätigkeit. Weiterhin werden mitarbeiterspezifische Informationen darin aufbewahrt, die unter anderem als Default-Werte in Formularen erscheinen können, um dem Nutzer Schreibarbeit zu ersparen. Das Mitarbeiterverzeichnis wird als ein XML-Dokument angelegt und entspricht derzeit folgender Dokumenttyp-Definition:

```
<!ELEMENT Mitarbeiterverzeichnis (Mitarbeiter*)>
<!ELEMENT Mitarbeiter            (Anrede,Vorname,Nachname,Kürzel,Telefon,
                                  Raum,Email,Unterschriftszeile1,
                                  Unterschriftszeile2,Zugriff,Admin)
<!ATTLIST Mitarbeiter            ID                    ID         #REQUIRED>
<!ELEMENT Anrede                 (#PCDATA)>
<!ELEMENT Vorname                (#PCDATA)>
<!ELEMENT Nachname               (#PCDATA)>
<!ELEMENT Kürzel                 (#PCDATA)>
<!ELEMENT Telefon                (#PCDATA)>
<!ELEMENT Raum                   (#PCDATA)>
<!ELEMENT Email                  (#PCDATA)>
<!ELEMENT Unterschriftszeile1    (#PCDATA)>
<!ELEMENT Unterschriftszeile2    (#PCDATA)>
```

```
<!ELEMENT Zugriff>
<!ATTLIST Zugriff               Angebotseinholung     (ja|nein)  "nein"
                                Beschaffungsantrag    (ja|nein)  "nein"
                                Brief                 (ja|nein)  "nein"
                                Fax                   (ja|nein)  "nein"
                                Gutachten             (ja|nein)  "nein"
                                Hausmitteilung        (ja|nein)  "nein"
                                Zugangsberechtigung   (ja|nein)  "nein">
<!ELEMENT Admin>
<!ATTLIST Admin                 value                 (ja|nein)  "nein">
```

Einträge für mitarbeiterspezifische Informationen können jederzeit verändert werden, indem der Administrator in der Datei `header.tcl` die Liste `defaultList` editiert und Elemente löscht oder hinzufügt. Die Einträge zur Rechtedefinition für Dokumententypen werden automatisch bei erfolgreicher Verarbeitung eines neuen Schemas erweitert (ist an `root.tcl` geknüpft) und als Default-Wert auf „nein“ stellvertretend für keine Berechtigung gesetzt. Außerdem gibt es einen statischen Eintrag für das Administratorrecht. Der Systembetreuer muss allen zutreffenden Mitarbeiter im Mitarbeiterverzeichnis den Zugriff auf einen neuen Dokumententyp erlauben. Dies ist natürlich aufwendig, wenn alle URZ-Mitarbeiter das Zugriffsrecht erhalten sollen, kommt aber sicherlich nicht oft vor und ist deshalb vertretbar. In der Eingabemaske zur Manipulation des Mitarbeiterverzeichnisses (Abbildung 5.9/(b)Nutzerverwaltung) erhält der Administrator die Möglichkeiten, einen neuen Nutzer einzutragen, Benutzerdaten zu bearbeiten oder den Benutzer aus dem Verzeichnis zu entfernen und nach erledigten Änderungen das Mitarbeiterverzeichnis abzuspeichern. Alle aufgeführten Funktionen sowie das Übernehmen von neuen Daten eines Mitarbeiters basieren wiederum auf der Änderung eines DOM-Baumes, der für `user.xml` angelegt wird. Das Mitarbeiterverzeichnis wird in einem separaten CGI-Ablauf manipuliert, bei dem die einmal gelesene XML-Mitarbeiter-Struktur als serialisierte xml-Variablen exportiert (hidden) wird. Vor Beginn aller möglicher Aktionen (mit Selektierung des Buttons Mitarbeiterverzeichnis) wird die Datei `user.xml` geparst oder, falls nicht vorhanden, angelegt. Die Funktion „Neuer Eintrag“ und „Mitarbeiter bearbeiten“ basieren auf einem Formular, das wie oben erwähnt aus `root.tcl` und der Liste `defaultList` generiert wird, also auch dynamisch ist. Die Eingaben werden in die XML-Strukturen des Mitarbeiterverzeichnisses eingefügt.

Eingabehilfe für Struktureinheiten

Zur einfachen und schnellen Eingabe von Struktureinheiten in Hausmitteilungen und Gutachten wurde eine Eingabehilfe zur Selektierung von Struktureinheiten per Auswahlliste integriert. Nach der Auswahl eines Eintrages wird eine JavaScript-Funktion aufgerufen, die die Daten der selektierte Zeile in die dafür vorgesehenen Eingabefelder übernimmt. Den Inhalt der Auswahlliste kann der Administrator bestimmen, indem er in der Eingabemaske „Editieren der Struktureinheiten“ die Textdatei `facprof.txt` editiert. Die Datei wird in einem Textarea angezeigt und bei Speicherwunsch als Übergabeparameter `text` an das `eoffice.cgi` übergeben. Dieser Request überschreibt die alte Datei mit den geänderten Daten.

6.4.4 Dokumentenfunktionen

Um ein Dokument zu manipulieren, wird es dem Programm als DOM-Baum verfügbar gemacht. Hierfür gibt es drei Varianten. Wenn ein Dokument vom Nutzer wie in Abschnitt 6.4.2 dargestellt geöffnet wird, wird das Dokument geparst. Ein neues Dokument wird sofort durch ein neues XML-Dokument im DOM repräsentiert. Falls keine der beiden Fälle auftreten, ist das XML-Dokument serialisiert im Übergabeparameter `xml` vom HTML-Formular übergeben worden und wird geparst. Bei allen Methoden stellt TclDOM den ersten Knoten des XML-Dokumentes bereit, der der Wurzel des DOM-Baumes entspricht. Während der Bearbeitung des Dokumentes ausführbare Funktionalitäten sind Speicher- und Signierfunktionen, die folgend angesprochen werden.

Digitale Signatur

Zum Nachweis von Integrität und Authentizität kann eine digitale Signatur auf das aktuell in der Bearbeitung befindliche Dokument erstellt werden. Dazu wird das Paket `gnupg` eingesetzt, dessen Einrichten dokumentiert ist (Abschnitt 7.1). Wie im Entwurf erläutert, wird zum Signieren und Verifizieren eine Methode gewählt, die der Situation angemessen ist. Der Webserver-Account (im Default-Fall `nobody`) verwendet ein Schlüsselpaar und kann damit Dokumente signieren und verifizieren. Zugleich ist der authentifizierte Nutzer bekannt, der den Vorgang veranlasst. Nutzer und Signatur werden in einem XML-File gespeichert, das folgender DTD entspricht.

```
<!ELEMENT  Signatures  (Signature)+>
<!ATTLIST  Signatures  Id ID #REQUIRED>
<!ELEMENT  Signature   (User,SigValue)>
<!ELEMENT  User        (#PCDATA)>
<!ELEMENT  SigValue    (#PCDATA)>
```

Wenn das geöffnete Dokument von anderen Nutzern gelockt ist, sollte der Mitarbeiter vor einer Signatur die anderen Bearbeiter kontaktieren (tritt im Regelfall sowieso ein), da kein weiterer Schutz gegen einen eventuellen mehrfachen Schreibzugriff auf das Signatur-File vorgesehen wurde. Beim Signieren wird durch

```
gpg --homedir $gpgDir -b -a Geschäftsdokument.xml
```

eine ASCII-Signatur erstellt, die inklusive des Nutzers in eine XML-Datei (durch '.sig' gekennzeichnet: `'Geschäftsdokument'.sig.xml`) aufgenommen wird. Jedes weiter Signieren erzeugt eine Signatur, die dem File hinzugefügt wird. Beim Verifizieren werden alle in der Signatur-Datei vorkommenden Signaturen nacheinander separiert und unter Verwendung von `gpg` mit der Option `--verify` verifiziert. Der Nutzer erhält eine Nachricht über die Korrektheit der Signatur.

Weitere Dokumentenfunktionen

Zu jeder Dokumentenbearbeitung gehören einfache Funktionen, wie Speichern, Speichern als, Löschen und Schließen von Dokumenten. (Abbildung 5.10) Speichern und Löschen benötigen ein Arbeitsverzeichnis, das aus einem Dokumentenpfad, einem Dokumententyp und einem Jahr besteht. Der Typ ergibt sich aus dem ersten Element der Tcl-Liste `path`. Das Jahr ist im Dateinamen enthalten, der bei der Neuerstellung definiert und als Attribut des Root-Elementes angelegt wird. Solange keine Datei mit diesem Namen existiert, bleibt die Funktion Speichern und Löschen inaktiv. (ebenso Signieren und Verifizieren) Zur Generierung eines vollständigen Namens muss nach dem Selektieren von Speichern als in ein JavaScript-Eingabefenster eine maximal 15 Zeichen lange Zeichenkette bestehend aus alphanummerischen Zeichen und Unterstrichen eingegeben werden. Demnach ergibt sich der vollständige Name eines Dokumentes, der in jedem Fall durch eine Testfunktion eindeutig generiert wird, aus Nutzer-String, Nutzerkennzeichen,Tag und Zeit. Wenn eine Datei mit einem solchen Namen schon existiert, kann diese überschrieben (einfaches Speichern) und gelöscht werden. Beim Schließen wird der Arbeitsbildschirm verlassen und ins Hauptmenü gewechselt. Wurde ein neu angelegtes Dokumenten nicht gesichert, erhält der Nutzer dabei eine Warnung und kann im Browser zum vorherigen HTML-Formular der Dokumentenbearbeitung zurückgehen, um das Speichern eventuell vorzunehmen. Die Lock-Datei wird beim Verlassen gelöscht. Falls ein Nutzer ein Dokument nicht schließt - einen anderen URL anwählt oder der Browser oder das System abstürtzt - bleibt das beim Öffnen angelegte Lock-File für eine Mindestdauer einer in `header.tcl` eingestellen Zeit erhalten, da das System keine Anweisung zum Entfernen erhält. Durch ein wiederholtes Öffnen nach dieser Zeit entfernt das Programm das alte Lock-File automatisch.

6.4.5 Interne Verarbeitung

Aus in Abschnitt 6.1.2 aufgeführten Grund (Verhalten von TclDOM) werden Zeichen mit Sonderbedeutung vor der internen Verarbeitung ersetzt. Dies sind:

```
<     ;lt;
>     ;gt;
"     ;quot;
&     ;amp;
'     ;apos;
_     ;uline;
\     ;back;
```

Die ersten fünf sind Sonderzeichen aus HTML, die normal als Entitäten in Dokumenten erscheinen. Unterstriche werden im Text ersetzt, da Attributnamen auch Unterstriche enthalten dürfen, die aber bei einer Vorbereitung der Konvertierung eines XML-Dokumentes in eine „LaTeX-fähige“ XML-Version nicht durch _ ersetzt werden dürfen, wie es mit den im Text vorkommenden Unterstrichen geschieht. Backslashs sind LaTeX-Steuerzeichen und wurden zur Vereinfachung und Vermeidung von Konvertierungsproblemen auch substituiert.

Die interne Verarbeitung kann in drei Teile gegliedert werden.

- Übernehmen der Eingabedaten in XML-Strukturen
- Manipulation der Pfade
- Auslesen von XML-Daten zur Formulargenerierung

Ein Nutzer gibt bei der Navigation zwischen verschiedenen HTML-Formularen ständig Variablen für Text, Attribute oder Listen mit oder ohne Inhalt an das CGI zurück. Weiterhin werden ein Pfad aus Elementen und ein Pfad der entsprechenden Positionsnummern geliefert. Beide beeinflussen die interne Verarbeitung.

Integration von Formulardaten in XML-Strukturen

Vor dem eigentlichen Eintragen der Daten werden vom Programm zwei Aspekte untersucht. Es wird festgestellt, ob

1. Daten als Text, Listen oder Attribute übergeben wurden.
2. der übergebenen Pfad im DOM-Baum existiert.

Daraus ergeben sich die vier aufgelisteten für die Integration der Daten relevante Möglichkeiten.

1. keine Daten, kein Pfad: keine Arbeit
2. keine Daten, Pfad vorhanden: Daten löschen
3. Daten vorhanden, kein Pfad: Pfad anlegen, Daten eingliedern
4. Daten vorhanden, Pfad vorhanden: Daten aktualisieren oder eingliedern

Nach dem Testen des Vorhandenseins von übermittelten Formulardaten wird der benötigte Knoten zur Manipulation von XML anhand der übergebenen Listen `path` und `posidpath` in einem Durchlaufen des DOM-Baumes gesucht. Sind Daten übergeben worden, wurde aber kein Knoten gefunden (Pfad exsistiert nicht), dann wird dieser angelegt und die Daten eingetragen. Beim Bestehen des Pfades besteht die Möglichkeit, dass entweder schon Inhalte existieren, die zu aktualisieren sind oder dass neue Elemente und Attribute in der XML-Struktur erzeugt werden müssen. Falls der Pfad existiert, aber keine Daten geliefert werden, bedeutet dies das Entfernen der vorhandenen Attribute und des Textes.

Bei mehrfach vorkommenden Elementen wird die Position des durch den aktuellen Knoten repräsentierten Elementes mit der entsprechenden Position im `posidpath` verglichen. Einfügen und Löschen von Text wird für einfach oder mehrfach vorkommende Elemente unterschiedlich bearbeitet. Das Eingliedern von Daten aus Listen, wie der Adressliste, der Mitarbeiterliste oder gelisteten Checkbox-Einträgen wird separat durchgeführt. Dabei werden weitere Elemente (komplexerer Typen) und Attribute im DOM-Baum erzeugt und mit Inhalt gefüllt.

Pfad ändern

Nach dem Eintragen der übergebenen Daten werden Vorbereitungen zur Erstellung eines neuen Formulars getroffen. Der Nutzer kann zwischen HTML-Formularen navigieren und verschiedene Pfade ansteuern. Wenn er sich tiefer in der DOM-Hierarchie abwärts bewegt, wird der Pfad um den übergebenen Elementennamen (`$submit`) erweitert. (Die Position wurde schon beim Erzeugen erweitert.) Beim Zurückgehen wird der Pfad um ein Level gekürzt. Dies betrifft auch den Positionpfad, wenn für diesen Level eine Position eingetragen war. Es ist auch möglich, dass im Dokument nur navigiert wird, ohne Elemente anzulegen. Dann wird nur der Pfad erstellt und reduziert, ohne `posidpath` zu ändern. Die Pfad-Elemente selbst sind nicht als Link zur Selektion gewählt, da dies in der Änderungsansicht möglich ist.

Daten aus XML lesen für neues Formular

Nachdem der Pfad geändert wurde, soll der Knoten im Baum gefunden werden, der die Darstellung des HTML-Formulars bestimmt. Wie schon erwähnt, ist einer der Eckpunkte des Systems, dass für jedes Element in XML ein eigenes Formular generiert wird. Das Selektieren des Knotens erfolgt etwa nach dem selben Prinzip wie beim Einfügen der Daten. Das Durchlaufen des Baumes und Selektieren des Pfades kann auch einfacher durch Funktionen geschehen, die in XPath 1.0 [XPath] implementiert sind, die aber die verwendeten Werkzeuge nicht unterstützten. Der Text und die Attribute werden ausgelesen (soweit sie existieren) und in Variablen abgelegt. Dies geschieht auch für die Auswahllisten Adresse/Mitarbeiter und für die Checkboxen. Diese Variablen stehen nachfolgend der Formularerzeugung zur Verfügung.

6.4.6 Dynamische Formulargenerierung

Die Eingabemaske der Dokumentenbearbeitung wird mit HTML-Tabellen aufgebaut. (Abbildung 5.10) Nach der Informationsbox für Systemnachrichten an den Nutzer (z.B. Verifizierstatus, Speichern) folgt das eigentliche Eingabe-Formular. In der ersten Zeile wird der Pfad dargestellt, der auch die jeweiligen Positionen enthält. Dieser wird anhand von `path` und `posidpath` ermittelt. Falls es für einen Eintrag in `path` keine Entsprechung in `posidpath` gibt, dann handelt es sich um ein neues Element, das auch dementsprechend mit „neu“ gekennzeichnet wird. Die darauffolgende Zeile von Navigations-Buttons ergibt sich aus der definierten Formularstruktur. Eine Erleichertung der Bearbeitung eines Dokumentes stellt ein „Neu“-Button dar, der wie die anderen auch das Übernehmen der Daten veranlasst. Der Pfad wird aber nicht geändert und nur der `posidpath` um ein Level geürzt, sodass das entsprechende Element wiederholt angelegt werden kann. (z.B. das mehrfache Anlegen von Listen-Elementen oder Textabschnitten) Der „Neu“-Button wird auf Grund des Elemententyps `*` angezeigt, der mehrfach vorkommendes Element verdeutlicht.

Danach werden die Attribute als einzeilige Eingabefelder aufgelistet. Die Namen werden der definierten Formularstruktur entnommen und mit den ausgelesenen Attribut-Variablen des XML-Dokumentes gefüllt. Für einige Attribute (und Elemente) sind Default-Werte im Mitarbeiterverzeichnis oder allgemein in `header.tcl` definiert, die das Programm automatisch einsetzt, wenn deren Inhalt leer ist. In einem Sonderfall erscheint wie in Abschnitt 6.4.3 angesprochen eine einzeilige Auswahlliste, die eine JavaScript-Funktion zur Eingabehilfe von zwei Attributen aufruft.

Nach den Attributen werden die HTML-Entsprechungen der komplexen Elementtypen angezeigt, welche in Abschnitt 6.2.2 aufgeführt wurden. Der einfache Typ entspricht normalem Text, der durch ein Textarea eingegeben werden kann. Als weitere Hilfen zur Eingabe sind Checkbox- und Radiobuttons implementiert. Die benötigten Default-Werte werden der Formularstruktur entnommen und können gegen ausgelesene Optionen geprüft werden. (selected)

Weiterhin werden Auswahllisten für Firmenadressen und Mitarbeiterdaten angeboten. Zur Anzeige von Mitarbeiterdaten wird die Datei `user.xml` geparst, alle in der Formularstruktur definierten Elemente ausgelesen und in einer Auswahlliste aufgeführt. Bereits selektierte und im XML-Dokument vorhandene Einträge werden auch beim Ändern wieder selektiert. (ebenso bei der Adressliste)

Anbindung an die Adressdatenbank: Um die Eingabe von Firmenadressen einfacher zu gestalten, werden alle möglichen Empfängeradressen aus einer Datenbank in einem Auswahlfeld zur mehrfachen Selektierung angezeigt. Im System wird zur Anbindung an die MySQL-Adressdatenbank des URZ die Tcl-Erweiterung `mysqltcl` verwendet. Sie bietet eine sehr einfache Schnittstelle, um auf MySQL-Datenbanken zuzugreifen. Vor der Verwendung wird das Paket dynamisch geladen. (libmysqltcl.so) Die erforderlichen Parameter zum Datenbankzugriff (Datenbankserver, Datenbank, Tabelle, Nutzer, Lesepasswort) sind in `header.tcl` definiert. Alle in der Datenbank befindlichen Einträge werden auf in der Formularstruktur definierten Werte abgefragt. Diese legt das Programm als Attribute im XML-Dokument an, die auch im Schema als solche definiert werden müssen. Bei mehreren selektierten Adressen werden demnach mehrere Adresselemente angelegt.

Als Abschluss der elementbezogenen Eingabemaske erscheint eine Button-Leiste, die die Namen „Zurück/Übernehmen“, „Ändern“ und „Löschen“ beinhalten kann. „Zurück/Übernehmen“ veranlasst das System, die Daten in XML zu übernehmen und den Pfad um einen Level zu kürzen, falls nicht schon das Wurzel-Element erreicht ist. Die beiden letzteren erscheinen allerdings nur, wenn das Formular über die Änderungsansicht aufgerufen wurde.

6.4.7 Dokumentenbearbeitung

Die Bearbeitung der Dokumente erfolgt durch die Eingabe von Text in die im Formular enthaltenen Eingabefelder. Die Daten werden dann zusammen mit weiteren Variablen an das CGI gesandt und im XML-Dokument aktualisiert, neu eingetragen oder gelöscht. Nicht alle eingegebenen Daten können durch einfaches Navigieren im Dokument erreicht werden. Wie schon angesprochen gibt es Elemente, die mehrmals vorkommen können. Um diese Einträge editieren zu können, wird die folgend vorgestellte Technik angewandt.

Änderungsansicht

Beim Navigieren durch die dynamisch generierten HTML-Formulare eines Dokumentes ist es nicht möglich, mehrfach vorkommende Elemente darzustellen. Dies soll am Beispiel einer Liste verdeutlich werden, die in der Formularstruktur mehrere untergeordente Elemente enthält. Da immer die Möglichkeit bestehen muss, ein neues Element der Liste anzulegen, kann das entsprechende HTML-Formular kein vorhandenes Element durch normale Button-Navigation anzeigen. Es wäre auch nicht bestimmt, welches angezeigt werden soll. Bei einfach vorkommenen Elementen kann dieses zum Ändern durch einfaches Navigieren im Dokument herausgelesen werden. Hier bietet die Änderungsansicht Abhilfe, die direkt für den Zweck der Änderung vorgesehen ist. Des Weiteren kann man durch sie einen besseren Überblick über die aktuelle Struktur und den Bearbeitungsstand des Dokumentes erhalten. Falls das Anzeigen der Änderungsansicht aktiviert ist, wird das aktuelle in DOM repräsentierte Dokument rekursiv durchlaufen und in strukturiert eingerückter Form unter dem Eingabeformular angezeigt. Dabei werden Elemente und Text als HTML-Link dargestellt. Der Link enthält einen JavaSript-Aufruf und die Parameter `submit, path, pathposid` und `node`. Dieser Node referenziert einen Knoten im DOM-Baum, der dadurch direkt angesprochen werden kann, ohne ihn erst durch rekursives Durchlaufen des Baumes suchen zu müssen. Durch Selektierung des Links wird per JavaScript das Laden eines neuen Formulars veranlasst, in dem die selektierten Werte angezeigt werden und zur Manipulation bereit stehen. Die schon erwähnten Buttons „Ändern“ und „Löschen“ erfüllen ihre durch den Namen vorgegebenen Funktionen, indem einerseits die Werte des Formulars in XML aktualisiert werden und der Pfad konstant bleibt und andererseits das XML-Element im Dokument entfernt wird.

Für die Typen Mitarbeiter und Adressen werden zur Speicherung der Inhalte Attribute generiert. Erstens können dadurch alle notwendigen Daten diese Bereiche zusammen in einem Formular als einzeilige Texteingabefelder angezeigt werden. Der zweite Punkt betrifft die Darstellung der Daten in der Änderungsansicht, die dadurch übersichtlicher gestalltet wird, da alle

Attribute in einer Zeile aufgeführt werden. Elemente hingegen stellen einen eigenen Hierarchie-Level dar (soweit sie nicht mehrfach vorkommen) und beanspruchen eine Zeile der Änderungsansicht. Diese erscheint beispielsweise für Serienbriefe sehr unübersichtlich, wenn alle Adress-Teile für mehrere Adressen untereinander aufgeführt würden. Dies steht etwas im Wiederspruch zur XML-Konvention, dass nur Eigenschaften von Elementen als Attribute definiert werden sollten. In diesem Fall behindert diese Definition den weiteren Umgang mit den XML-Dokumenten nicht und dient der Benutzerfreundlichkeit des Systems.

6.4.8 Konvertierungen

HTML-Preview

Die HTML-Voransicht dient dazu, dass sich der Nutzer schnell einen Überblick über den Bearbeitungsstand und die fertige Formatierung im Zieldokument ansehen kann. Dazu wird ein neues Browser-Fenster mit einer HTML-Datei geöffnet, die durch eine XSLT-Engine aus dem XML-Dokument und einem XSL-Stylesheet erzeugt wurde. Für die Konvertierung wird mit `mktemp` ein temporärer Dateiname erzeugt. Zur Ablage der erstellten Dateien dient das `/tml`-Verzeichnis des Servers. Nach der Aufbereitung des XML-Dokumentes für die Konvertierung (Substitution der ersetzten Zeichen)erfolgt die Umwandlung durch die XSLT-Engine Xalan, die den XML-Parser Xerces zum Einlesen von XML- und XSL-Dokumente verwendet. Die Einbindung der Engine basiert auf einem Kommandozeilenbefehl und ist damit einfach eingegliedert und austauschbar. Die gesamte Konvertierung geschieht unabhängig von der weiteren Bearbeitung des geöffneten XML-Dokumentes. Verwendete Stylesheets werden aus einem Server-Verzeichnis der hochgeladenen Formulardaten gelesen. Nach der Transformation wird die HTML-Datei an das neue Browser-Fenster gesendet. Alle temporäre Datei werden gelöscht. Zur Erzeugung von XSL-Styles sind notwendige Informationen aus der Dokumentation (Abschnitt 7.2.2), der „XML-Bible“ [Har00] und der XSLT-Spezifikation [XSLT] zu erfahren.

PostScript-Generierung

PostScript ist das allgemein verwendete Druckformat im URZ. Bei der Darstellung von PostScript muss darauf geachtet werden, dass der Header „Content-type: application/postscript“ enthält und die Länge des Dokumentes in „Content-lenght“ angegeben wird. Temporäre Dateinamen werden wie bei der HTML-Konvertierung mittels `mktemp` generiert. Nach einer Zeichenkonvertierung für LaTeX-Sonderzeichen kommt wiederum Xalan zum Einsatz. Dabei wird das aufbereitete XML-Dokument durch die Verwendung eines Stylesheets in eine Textdatei konvertiert, die LaTeX-Befehle enthält. Dieses wird durch LaTeX in eine `dvi`-Datei übersetzt, die im Anschluss mittels `dvips` nach PostScript konvertiert wird. Das Fax wird zweimal übersetzt, da sich darin eine LaTeX-generierte Seitenangabe befindet. Bei der Übersetzung der LaTeX-Dokumente werden LaTeX-Dokumentenklassen verwendet, die im Zuge der Diplomarbeit von Herrn Dr. Riedel neu überarbeitet bereitgestellt wurden. Alle erzeugten temporären Dateien werden nach dem Senden des generierten PostScript-Dokuments an den Browser gelöscht. Darunter fallen auch die bei der LaTeX-Übersetzung erstellten Dateien. Zur Ansicht des PostScript-Dokumentes ist in der Browser-Konfiguration eine Anwendung zur Interpretation von PostScript-Inhalten zu wählen (`gv` empfehlenswert). Zur Definition der XSL-Stylesheets sind die selben Quellen wie beim HTML-Preview ratsam. Deren Erstellung wird in Abschnitt 7.2.2 angesprochen.

6.5 Zusammenfassung

In diesem Kapitel wurden implementierte Funktionalitäten des eOffice beschrieben. Die Implementation der auf XML beruhenden Verarbeitung basiert auf folgenden Eckpunkten:

1. Ein Dokument wird in mehreren HTML-Formularen erstellt, zwischen denen navigiert werden kann.

2. Der Aufbau von HTML-Formularen wird durch eine Struktur bestimmt, die aus Tcl-Feldern für Elemente, Elementtypen und Attributen besteht.

3. Jedes XML-Element veranlasst das Erzeugen eines eigenen HTML-Formulars.

4. XML-Attribute werden in HTML-Formularen als einzeilige Eingabefelder dargestellt; XML-Elemente können verschiedene Typen, respektive HTML-Konstrukte repräsentieren.

5. Die Implementation bürgt für die korrekte wohlgeformte Erzeugung von XML-Dokumenten.

Zur CGI-Programmierung wurde das Paket `cgi.tcl` von Don Libes genutzt, das eine beachtliche Erleichterung brachte. Die intere Verarbeitung basiert auf XML und wird vorrangig durch das Paket TclDOM unterstützt. Sie geschieht in dem CGI-Skript `eoffice.cgi`, welches dazu wiederholt aufgerufen wird.

Die Implementation des eOffice beinhaltet Interfaces und Funktionen für Nutzer und Administratoren. Während der Systembetreuer über ein Web-Interface das Nutzerverzeichnis verwaltet oder das eOffice um neue Dokumente erweitert, steht aus Nutzersicht die Dokumentenerstellung im Vordergrund. Dazu sind Funktionalitäten üblicher Textsatzsysteme implementiert worden (z.B. Öffnen, Speichern) und Zusätze wie das Erstellen von digitalen Signaturen. Durch eine Mail-Funktion wird dem Nutzer ermöglicht, einen Link auf ein gespeichertes XML-Geschäftsdokument an eine Person zu mailen. Dafür wird ein kleines Browser-Fenster geöffnet, in dem ein Mail-Formular angezeigt wird. Weiterhin wird die Visualiserung der XML-Dokumente in HTML und PostScript unterstützt.

Kapitel 7

Dokumentation

7.1 Installation

Die Installation des Systems beinhaltet die nachfolgend aufgeführten Schritte und sollte auf einem separaten Server erfolgen. Nötige Software befindet sich auf der beiliegenden CD und die Quellenangaben in Abschnitt 5.2.2.

- Installation eines SSL-fähigen Apache-Web-Servers (Web-Server-Account mit eingeschränkten Rechten)
  - Anlegen eines Verzeichnisses `$cgi` zur Ausführung von CGI-Programmen oder Nutzen des vorhandenen `/cgi-bin` der Apache-Installation
  - Kopieren folgender Dateien nach `$cgi`
    - eoffice.cgi, html.cgi, postscript.cgi, mail.cgi,
    - header.tcl, proc.tcl, dom.tcl, xml.tcl, sgml.tcl
- Erstellen eines öffentlichen Web-Verzeichnisses `$js` zur Ablage von JavaScript
  Kopieren von eoffice.js nach `$js`
- Erstellen eines öffentlichen Web-Verzeichnisses `$img` zur Ablage von Bildern
  Kopieren aller Image-Buttons und Logos des eOffice nach `$img`
- Erstellen eines Verzeichnisses `$docs` zur Ablage von eOffice-XML-Dokumenten (Zugriffsrechte nur für den Web-Server)
- Erstellen des Verzeichnisses `$data` zur Ablage von eOffice-Daten (Zugriffsrechte nur für den Web-Server)
- Erstellen der zu `$data` untergeordneten Verzeichnisse `$data/.sessions` und `$data/.gnupg` (`$gnupg`)

- Installation von Tcl (mindestens Version 8.2.0)
- Installation von Java und Bereitstellung von xalan.jar und xerces.jar; Umgebungsvariable `CLASSPATH` setzten (oder einfach beide Pakete nach `$cgi` kopieren)
- Installation von `mysqltcl` und `gnupg`
- Erzeugung eines Schlüsselpaares: (Zugriffsrechte nur für den Web-Server)
  ```
  gpg --homedir $gnupg --gen-key
  ```
  Es folgen Fragen zur Art der Schlüssel, Schlüsselgröße und Gültigkeitsdauer. Weiterhin sind Angaben zur Nutzer-Id zu tätigen. (Der Name wird als Ersteller der Signaturen im

eOffice angezeigt.) Weitere Informationen sind durch `man gpg` zu erhalten. Die Abfrage des Pass-Phrase muss allerdings zweimal durch Enter bestätigt werden ohne Angabe eines Zeichens (leerer Pass-Phrase).

- Setzen von (kommentierten) Tcl-Variablen in `header.tcl` (erstellte Verzeichnisse und Pfade der installierten Programme)

7.2 Systemadministration

7.2.1 Administratortätigkeiten

Wenn ein Mitarbeiter das Administratorrecht erhält, wird er im Hauptmenü einen Button `Administration` vorfinden, der ihn in das Administratormenü führt. Im Administratorbereich des eOffice sind die drei Funktionen

- Integration neuer Geschäftsdokumente,
- Nutzerverwaltung,
- Bereitstellen einer Liste von Struktureinheitenbezeichnungen

als Webschnittstelle verfügbar gemacht worden, die im Weiteren erläutert werden.

Integration neuer Geschäftsdokumente

Um die ständige Erweiterung der zu erstellenden Dokumententypen zu gewährleisten, kann der Administrator neue Formularstrukturen zur Verfügung stellen.

Unter dem Menüpunkt *Formularstruktur(en)* erscheint im Arbeitsbereich eine Eingabemaske zum Hochladen von drei Dateien. Das sind

1. ein Schema zur Generierung der Formularstruktur,
2. ein XSL-Stylesheet zur Konvertierung von XML nach HTML,
3. ein XSL-Stylesheet zur Konvertierung von XML nach LaTeX.

Diese vom Systembetreuer anzulegenden Dateien werden benötigt, um die Erzeugung von Geschäftsdokumenten realisieren zu können. In Abschnitt 7.2.2 ist die Erzeugung der Dokumente dokumentiert.

Nutzerverwaltung

Zur Verwaltung der Nutzer des Systems besitzt der Administrator die Möglichkeit, Daten der Mitarbeiter im Mitarbeiterverzeichnis anzulegen, zu bearbeiten oder zu löschen. Die gespeicherten Daten dienen der Rechtevergabe und der Eingabehilfe. Wie ABB zeigt, gibt es folgend angeführte offensichtliche Funktionen.

- Erstellen eines neuen Nutzereintrages mittels des Buttons „Neuer Eintrag“.
- Auswahl vorhandener Nutzer anhand des Nutzerkennzeichens in einer Auswahlliste.
- Manipulation der Daten des selektierten Nutzers durch „Bearbeiten“.
- Löschen des Eintrages des selektierten Nutzers.
- Speichern des Mitarbeiterverzeichnisses nach Beendigung der Aktualisierung.

Nach dem Eintragen neuer Daten muss dies mit dem Button „Mitarbeiterdaten übernehmen“ bestätigt werden.

Bereitstellen einer Liste von Struktureinheitenbezeichnungen

Eine weitere Eingabehilfe wird durch das Anlegen einer Struktureinheitenliste unterstützt, die der Administrator verwaltet. Darin sollten Struktureinheiten aufgenommen werden, die in der Hausmitteilung und im Gutachten häufig benutzt werden. Ein Eintrag muss dabei einer der beiden folgenden Konventionen entsprechen.

- Angabe einer Einheit (z.B. Fakultät)
- Angabe einer Einheit und einer untergeordneten Einheit; durch ein Komma getrennt (z.B. Fakultät, Professur)

Im Formular werden diese Einheiten als Fakultät und Professur bezeichnet, da dies den häufigsten Fall darstellt. Die beiden Bereiche können auch Fakultät/Institut oder Dezernat/Abteilung entsprechen.

Falls Systemfehler auftreten sollten hat der Administrator die Möglichkeit, eine Log-Datei einzulesen. Schwere Fehler lösen das Versenden einer Email an die durch `cgi_admin_mail_addr` in `header.tcl` gesetzten Adresse aus. Bei Einschalten des Debug-Modus durch `cgi_debug -on` in `header.tcl` werden Debug-Informationen und eine ausführliche Fehlermeldung während der Programmausführung am Bildschirm angezeigt, die den Fehler in der Regel schnell erkennen lassen.

7.2.2 Erstellen benötigter XML-Dokumente

Erstellen von Schemata

Zur Einbindung eines Geschäftsdokumentes werden von System Schemata verarbeitet und daraus Formularstrukturen zur Generierung von HTML-Formularen erstellt. Die folgenden Tabellen geben einen Einblick in die verwendeten Tags zur Erstellung der Schemata. Die Schemata sind nicht XML-Schema-konform [XML-Schema] , können aber auch dem Erstellen von W3C-konformen Schemata dienen. Das System beschränkt sich ausschließlich auf das Erzeugen von Formularstrukturen.

Element	Bedeutung
schema	Wurzel-Tag des Schemas
element	Führt ein neues Element ein, für das im eOffice entsprechend dem Typ ein eigenes Formular generiert wird. (Ausnahme Default-Elemente für CheckboxButtons)
attribute	Führt ein neues Attribut eines Elementes ein, das im eOffice im Formular des Elementes als einzeiliges Eingabefeld angezeigt wird.
type	Kennzeichnet eingebettete komplexe Typen. (untergeordnete Elemente und Attribute) - wird vom Programm ignoriert.

Tabelle 7.1: Elemente zur Definition von Schemata

Die durch die in den Tabellen aufgefühten Tags erzeugen Schemata wurden auch mit Schemata* bezeichnet. Deren Definition unterliegt folgender DTD, einem Subset von XML-Schema.

```
<!ELEMENT schema     (element)*>
<!ELEMENT element    (type)?>
<!ATTLIST element    name       NMTOKEN  #IMPLIED
                     ref        NMTOKEN  #IMPLIED
                     type       NMTOKEN  #IMPLIED
                     maxOccurs  CDATA    #IMPLIED
                     default    CDATA    #IMPLIED
                     fixed      CDATA    #IMPLIED>
<!ELEMENT type       (element,attribute)*>
```

Attribut	Bedeutung
name	Angabe eines Elementennamens; ausschließlich in alphanummerischen Zeichen ohne die Verwendung der Vokale ä,ü,ö (Ausnahme: Das erste Element kann die Namen mehrerer Dokumententypen (getrennt durch |) enthalten.
default	Standardtext, der zur Auswahl bereitgestellt werden soll (für Checkbox und Radiobutton)
ref	Verweis auf ein untergeordnetes Element
type	Type des Elementes (address, checkbox, facproflist, mitarbeiter, radiobutton, string)
maxOccurs	Definiert ein Mehrfachelement mit *
fixed	Bestimmt alle Dokumententypen (getrennt durch |), in deren Formularstruktur das Element dieses Attributes vorkommen soll.

Tabelle 7.2: Attribute von `element`

Attribut	Bedeutung
name	Angabe eines Elementennamens; (alphanummerischen Zeichen und Leerzeichen erlaubt, aber ohne die Verwendung der Vokale ä,ü,ö)
fixed	Bestimmt alle Dokumententypen (getrennt durch |), in deren Formularstruktur das Element dieses Attributes vorkommen soll.

Tabelle 7.3: Attribute von `attribute`

```
<!ELEMENT attribute>
<!ATTLIST attribute  name       NMTOKEN  #REQUIRED
                     fixed      CDATA    #IMPLIED>
```

In einer `<type>`-Umgebung ist das Attribut `name` des Elementes `element` nicht gestattet. Die Verwendung der Elemente und Attribute erfolgt in etwas abgeänderter Bedeutung wie in den Tabellen 7.1 bis 7.3 dargelegt.

Als kurzes Beispiel verdeutlicht das Schema der Zugangsberechtigung die Anwendung der DTD.

```
<?xml version="1.0"?>
<schema>

 <element name="Zugangsberechtigung">
  <type>
   <element ref="Mitarbeiterliste"      type="mitarbeiter"/>
   <attribute name="Datum"/>
   <attribute name="Gueltig bis"/>
  </type>
 </element>

 <element name="Mitarbeiterliste">
  <type>
   <element ref="Mitarbeiter"           maxOccurs="*"/>
  </type>
 </element>

 <element name="Mitarbeiter">
  <type>
   <attribute name="Anrede"/>
   <attribute name="Vorname"/>
   <attribute name="Nachname"/>
  </type>
```

```
 </element>

</schema>
```

Durch die Angabe des Types zu einem `ref`-Element (entsprechendes Attribut) erkennt ihn das verarbeitende Programm und legt die Formularstrukturen an. Das Element wird in weitere Teile strukturiert, die das Programm erwartet.

Für die im eOffice implementierten Typen gelten feste Bestimmungen. Für alle Objekte, die im HTML-Formular als Auswahllisten dargestellt werden, (Adressliste - `address`; Mitarbeiterliste - `mitarbeiter`) gilt folgende Konstruktion:

```
<element name="'Listenbezeichnung'">                          # z.B. Empaenger
 <type>
  <element ref="'Listenelement'"       maxOccurs='*'/>      # z.B. Adresse
 </type>
</element>
<element name="'Listenelement'">
 <type>
  <attribute name="'"/>
  # Aufzählung aller Teile eines Listenelmentes (z.B. Firma, Abteilung, usw)
 </type>
</element>
```

Checkboxen und Radiobutton erfordern Default-Einträge:

```
<element name="'Bezeichung der Auswahl'">
 <type>
  <element ref="Eintrag" type='string' maxOccurs='*'/> # nur für Checkboxen
  <element default="Auswahltext 1"/>
  <element default="Auswahltext 2"/>                   # beliebige viele Default-Elemente
 </type>
</element>
```

Erstellen von XSL-Stylesheets

Die bei der Konvertierung von XML-Dokumenten nach PostScript- oder HTML-Dokumenten verwendeten XSL-Stylesheets sind zum W3C Working Draft der XSL Version 1.0 vom 12. Januar 2000 konform [XSL] und in XSLT Version 1.0 [XSLT] nach der W3C Empfehlung vom 16. November 1999 implementiert.

Im folgenden Beispiel ist ein Ausschnitt eines HTML-Stylesheets zu sehen, der zur Konvertierung von XML-Dokumenten nach HTML verwendet werden kann.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html" encoding="ISO-8859-1"/>

  ...

  <xsl:template match="Betreff">
    <b><xsl:value-of select="."/></b>
  </xsl:template>

  ...

</xsl:stylesheet>
```

Template-Regeln werden durch das `xsl:template`-Element definiert und sind der wichtigste Teil von XSL-Stylesheets. Jedes dieser Elemente besitzt ein `match`-Attribut, welches spezifiziert, welcher Knoten des Eingabe-Dokumentes durch das Template instanziiert wird.

Im Falle des Beispiels wird der Inhalt des Knotens `Betreff` aus dem XML-Dokument

```
<Betreff>Betreffzeile</Betreff>
```

nach HTML konvertiert, indem das Element `xsl:value-of` durch das Attribut `select` das Einbetten des Knoteninhalts in HTML-Tags veranlasst.

```
<b>Betreffzeile</b>
```

Für die Transformation nach LaTeX werden die benötigten LaTeX-Befehle im Stylesheet erscheinen. Als weitere Anleitung zur Erstellung von XSL-Stylesheets soll auf den XSL-Bereich der „XML Bible“ [Har00] verwiesen werden.

7.3 Nutzerdokumentation

Das System stellt dem Nutzer zwei grundlegende Menüs zur Verfügung. Es sind der Anfangsbildschirm mit Hauptmenü und der Arbeitsbildschirm zur Erstellung von Dokumenten. Im Hauptmenü können Funktionen wie Auswahl eines Dokumententypes zur Neuerstellung und das Öffnen und Suchen von Dokumenten ausgeführt werden. Die Funktionalitäten des Arbeitbildschirmes umfassen weiterhin bekannte Aktivitäten aus Textsatzsystemen und systemspezifische Erweiterungen.

Zur Anzeige von erzeugten PostScript-Dokumenten muss in der Konfiguration einen Anwendung zu Interpretation von „application/postscript“-Inhalten gewählt werden.

7.3.1 Hauptmenü

Zur Erstellung eines neuen Dokumentes kann der Nutzer in der Eingabemaske einen gewünschten Dokumententyp selektieren. Er gelangt dann in den Arbeitsbildschirm und kann mit der Eingabe von Daten zur Erzeugung eines neuen Dokumentes des gewünschten Typs beginnen.

Um ein gewünschtes Dokument zu öffnen, sind in der Eingabemaske „Öffnen“ zuerst das Jahr, ein Nutzerkennzeichen und der Dokumententyp zu bestätigen. Dadurch wird eine Liste aller Dokumente des angegebenen Dokumententypes und Erstellungsjahres angezeigt, welche dem Nutzer des entsprechenden Nutzerkennzeichens zugeordnet sind. Zum Öffnen selektiert der Nutzer einen Dokumentennamen aus der Liste und bestätigt dies mit dem Button „Dokument öffnen“.

In der Eingabemaske Suchen sind außer dem Jahr und Dokumententyp auch noch ein Nutzerkennzeichen und ein Suchmuster zur Eingabe zugelassen. Bei Angabe eines Suchmusters wird dieses unabhängig von Gross- oder Kleinschreibung in den durch die anderen Angaben bestimmten Dokumenten gesucht. Alle Parameter ergeben auf einen Mausklick wiederum eine Liste von Dokumenten, von denen eines anschließend geöffnet werden kann.

7.3.2 Arbeitsmenü

Beim Neuerstellen und Öffnen eines Dokumentes wird in den Arbeitsbildschirm zum Erzeugen oder Bearbeiten von Geschäftsdokumenten gewechselt. Bevor auf das Erstellen und Bearbeiten detaillierter eingegangen wird, sollen die Menüfunktionen näher erläutert werden. Das Menü besteht aus mehreren Buttons, deren Funktionen folgend tabellarisch aufgeführt werden.

7.3.3 Arbeitsmaske

Die Arbeitsmaske besteht aus mindestens zwei und maximal vier Teilen (Abbildung 5.10). Das Informationsfeld erscheint, wenn eine Interaktion mit dem Nutzer erforderlich ist, wie zum Beispiel wenn eine Datei gelockt ist und nach dem Speichern, Signieren oder Verifizieren einer Datei. Im stets vorhandene zweiten Feld wird durch einen Pfad die Tiefe der Bearbeitung eines Dokumentes angezeigt. Das Kernstück des Systems ist das dynamische Formular, welches Daten zum Erstellen eines Dokumentes aufnimmt und eine Navigation durch das Dokument erlaubt. Die Änderungsansicht im vierten Teil kann wie beschrieben aktiviert und deaktiviert werden und präsentiert eine Baumstruktur des erzeugten Dokumentes.

Button	Funktionalität
Speichern	Speichern des aktuellen Dokumentes. (nur aktiv, falls das Dokument schon gespeichert wurde)
Speichern als	Speichern des aktuellen Dokumentes unter einem neuen Namen. (Das Dokument in der Bearbeitung ist nun dies mit dem neuen Namen.)
Löschen	Löschen des aktuellen Dokuments. (inklusive Signaturen und Lockdateien; nur aktiv, falls das Dokument schon gespeichert wurde)
Signieren	Signieren des aktuellen Dokuments im Auftrag des Nutzers.(nur aktiv, falls das Dokument schon gespeichert wurde)
Verifizieren	Verifizieren vorhandener Signaturen des aktuellen Dokuments. (nur aktiv, falls das Dokument schon signiert wurde)
Link senden	Mailformular zum Versenden eines Links auf ein gespeichertes Dokument.
Änderungsansicht	Anzeige einer Ansicht des aktuellen Dokumentes in Baumhierarchie mit Links zum Editieren des Inhaltes.
HTML-Voransicht	Öffnen eines kleinen Browserfensters mit einer HTML-Voransicht des aktuellen Dokumentes.
Postscript	Anzeigen des PostScript-Dokuments, das aus dem aktuellen Dokument generiert wurde. (in einem PostScript-fähigen Viewer)
Hilfe	Anzeigen einer Hilfeseite.
Schließen	Verlassen der Dokumentenerstellung und Wechsel in das Hauptmenü.

Tabelle 7.4: Buttonfunktionen der Arbeitsmenüs

Um das System korrekt verwenden zu können, ist es wichtig, zwei Punkte der Philosophie des Systems zu verstehen.

1. Ein Dokument wird hierarchisch aufgebaut. Um dies zu realisieren, werden die Dokumente in Teile zerlegt, die ihrerseits wiederum zerlegt werden.
   Beispiel: Der Brief besteht unter anderem aus den Teilen **Empfänger**, **Text** und **Gruß**. Der Text kann nun in die weiteren Teile **Abschnitt**, **Tabelle** und **Liste** zerlegt werden. Die Hierarchie endet, wenn eine weitere Zerlegung nicht mehr notwendig ist. Die Liste enthält nur noch **Elemente**, welche nicht weiter aufgesplittet werden.

2. Es existieren Elemente, die in einem Dokument nur einmal vorkommen, z.B. eine Betreffzeile oder eine Grußformel.
   Weiterhin gibt es auch Elemente, deren mehrmaliges Auftreten notwendig ist, z.B. Listenelemente oder Abschnitte in einem Text.
   Zwischen diesen Elementen bedarf es einer Unterscheidung, wie später näher ausgeführt wird.

Das Formular unterstützt die Navigation im Dokument und stellt dafür ober- und unterhalb der Eingabefelder Buttons zur Verfügung. Durch die Buttons oberhalb gelangt der Nutzer tiefer in die hierarchische Struktur hinein. Unterhalb der Eingabemöglichkeiten befindet sich der Button „Übernehmen/Zurück“, durch den die Eingaben übernommen werden und im Pfad ein Level zurückgesprungen wird. Als Arbeitserleichterung findet sich rechtsbündig oberhalb der Eingabefelder der Button „Neu“, der nur bei Elementen auftaucht, die mehrfach erzeugt werden können. Dieser erlaubt das Anlegen eines weiteren Elementes, ohne erst durch „Übernehmen/Zurück“

ein Level zurückzuspringen, um danach das selbe Element wieder zu selektieren. Beim Betätigen der Buttons oberhalb der Eingabemöglichkeiten werden die eingetragenen Formulardaten ebenso wie durch „Übernehmen/Zurück“ in das Dokument übernommen. Während der Neuerstellung erscheinen in einigen Eingabefeldern nutzerspezifische Defaultwerte, die einem Nutzerverzeichnis entnommen werden.

7.3.4 Änderungsansicht

Die Änderungsansicht zeigt eine Baumstruktur des erzeugten Dokumentes. Der primäre Grund für diese Ansicht ist das Bereitstellen von Links, um die dadurch referenzierten Daten in ein Formular laden und editieren zu können. Bei mehrfach vorkommenden Elementen ist dies durch einfache Navigation nicht möglich, weil dabei die Eingabemöglichkeiten ausschließlich für das Neuerstellen von Elementen bestimmt sind. Damit wird bereits angedeutet, dass Einträge einfacher Elemente durch die Navigations-Buttons erreicht und geändert werden können. Nach dem Laden der Daten in das Formular sind zwei weitere Buttons verfügbar. „Ändern“ aktualisiert den Datenbestand des Dokumentes bei gleichbleibendem Pfad. Durch das Selektieren des Buttons „Löschen“ wird das gesamte aktuell im Formular dargestellte Element entfernt. In der Änderungsansicht sind den Links X-Buttons vorangestellt. Damit wird es dem Nutzer ermöglicht, einen gesamten Zweig in der Baumstruktur des Dokumentes zu löschen.

Kapitel 8

Zusammenfassung und Ausblick

Zusammenfassung

Im Rahmen der Arbeit wurde ein modernes System zur Erledigung von Geschäftsvorfällen mit dem Namen „eOffice“ erschaffen. Es erweitert die Funktionalität des Intranets des Universitätsrechenzentrums um eine Applikation zur Erzeugung von Dokumenten. Zur Zeit können Geschäftsdokumente wie Briefe, Angebotseinholungen, Faxe, Hausmitteilungen, Gutachten und Zugangsberechtigungen über HTML-Formulare online erstellt werden. Diese werden dem Nutzer zur Voransicht in HTML und als druckbares Exemplar als PostScript zur Verfügung gestellt. Bei Bedarf ist es dem Systembetreuer jederzeit möglich, das System um neue Geschäftsdokumente zu erweitern. Ihm steht dazu und zur Ausführung weiterer Tätigkeiten ebenso eine Web-Schnittstelle zur Verfügung. Da das System zur Dokumentenerstellung verwendet werden soll, sind Grundfunktionen (Öffnen, Speichern, Suchen usw.) wie in anderen grafischen Textsatzsystemen enthalten. Weiterhin zeichnet es sich durch Funktionalitäten zur Unterstützung der Dokumentenerzeugung wie Serienbrieferstellung, Ausfüllhilfen, Auswahlmöglichkeiten, Datenbankanbindung und Default-Einträge aus. Zusätzlich können Dokumente signiert und verifiziert und ein Verweis auf das aktuell in Bearbeitung befindliche Dokument als Email versandt werden. Das System garantiert ein einheitliches Layout aller erzeugten Dokumente, welche auch in der Textgestaltung festen Normen unterliegen. Somit bleibt die Corporate Identity gewahrt. Dieser Prototyp ermöglicht eine effizientere Erstellung von Geschäftsdokumenten und rationalisiert damit den gesamten Geschäftsvorfall.

In der Erprobung wurde nach einiger Einarbeitungszeit die Funktionalität einer vereinfachten und schnelleren Erzeugung von Geschäftsdokumenten bestätigt. Schwierig gestaltet sich jedoch das Ändern vorhandener Geschäftsdokumente in Bezug auf die Änderung von Textpassagen, die unterschiedliche Formatierungen enthalten. Um diese zu editieren, kann auch das Löschen vorhandener Daten nötig sein, um sie in einer neuen Formatierung erneut einzutragen.

Ausblick

An dem entwickelten System können weitere Anpassungen und Erweiterungen vorgenommen werden. Dazu zählen:

- Anbindung des eOffice an das Faxsystem des URZ
- Einführung einer Typprüfung von Inhalten
- Verbessern der Funktionalität zur Textbearbeitung

Anbindung an das Faxsystem

Mit dem eOffice lassen sich Faxdokumente erzeugen, die im PostScript-Format zum Ausdrucken zur Verfügung gestellt werden. Um das Drucken der Faxe und das manuelle Bedienen des Faxgerätes zu vermeiden, kann das eOffice eine Funktionalität erhalten, die eine Kommunikation mit dem Faxsystem des URZ ermöglicht. Unter Verwendung von `hylafax` auf dem eOffice-Server können die Faxdokumente direkt aus dem eOffice versandt werden, wenn keine handschriftliche Unterschrift erforderlich ist.

Typprüfung von Datentypen

Das entwickelte System übernimmt alle Eingaben des Nutzers und führt keine Prüfung dieser durch. Nach der Erstellung eines Schemas, das die Anforderungen an Datentypen von Elementen und Attributen spezifiziert, können diese Datentypen zur Validierung der Eingaben verwendet werden. Dies könnte beispielsweise eine JavaScript-Funktion bei jedem CGI-Request übernehmen.

Textbearbeitung

Beim Schreiben von Text, z.B. im Textbereich von Briefen und Hausmitteilungen, werden die nacheinander erzeugten Textteile (z.B. Abschnitte) untereinander angelegt. Neu hinzugefügter Text wird nach dem letzten Teil eingeliedert. Ein Einfügen von neuen Abschnitten vor schon erzeugtem Inhalt ist zur Zeit nicht möglich. Das Level 1 des Document Object Model, auf dem die interne Verarbeitung beruht, bietet Methoden an, die eine solche Funktion unterstützen könnten.

Weitere Anpassungen könnten die Performance des Systems betreffen. Zum einen wurde das schon erwähnte Apache-Modul `mod_dtcl` seit kurzem offizielles Apache Software Foundation Projekt unter der Schirmherrschaft von Apache Tcl. Unter [ModTcl] kann der aktuelle Entwicklungsstand und die Dokumentation in Erfahrung gebracht werden. Die Nutzung des Moduls hat den Vorteil, dass der Umstieg von der aktuellen CGI-Programmierung zu `dtcl` relativ einfach realisierbar ist, da das Microscripting-Paket ein Teil des in der Implementierung verwendeten `cgi.tcl` darstellt. Falls die Performance des System noch erhöht werden sollte und `mod_dtcl` alle nötigen Funktionalitäten liefert, dann ist es durchaus eine Alternative zur CGI-Programmierung. Zum anderen werden auch für Tcl einfache XSL-Anbindungen [TclXSL] entwickelt, die die Nutzung der ressourcenraubenden Java-Pakete einsparen könnten, sich aber noch nicht in verwendbarem Zustand befinden.

Anhang A

Mitarbeiterverzeichnis des URZ

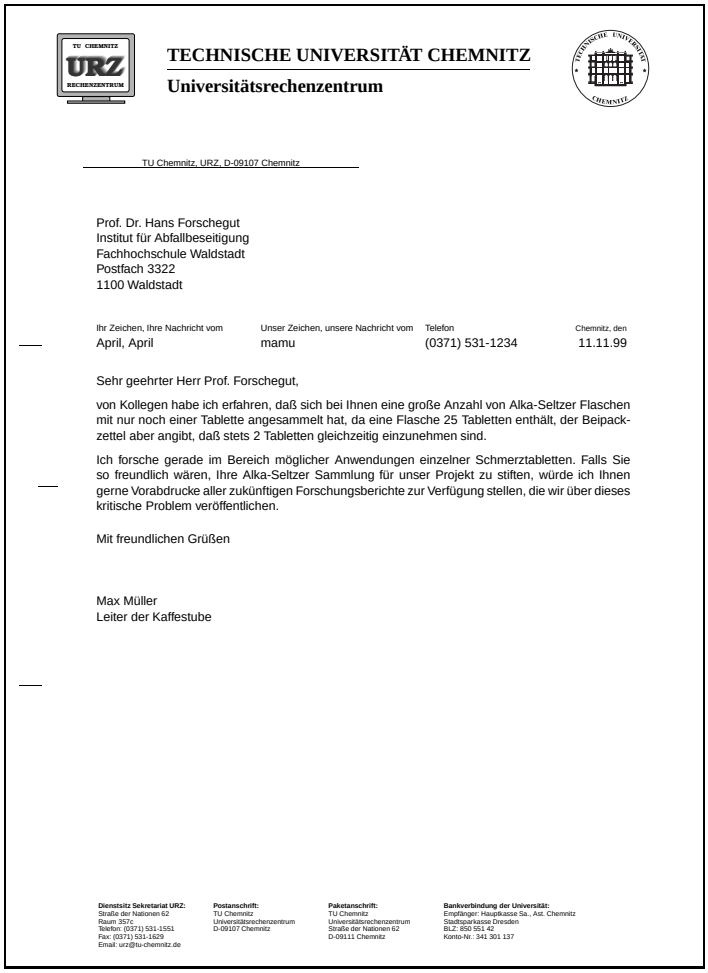
Name	Vorname	Kürzel	Telefon (531....)	Raum Str.d.Nationen 62	Email (...@hrz.tu-chemnitz.de)
Becher	Mike	mibe	1725	364a	M.Becher
Brose	Steffen	stb	1722	362	S.Brose
Clauß	Matthias	wmc	1468	363b	M.Clauss
Clauß	Udo	ucl	1428	361a	U.Clauss
Ehrig	Matthias	meh	1525	363a	M.Ehrig
Dippmann	Dagmar	ddi	1876	364c	D.Dippman
Fischer	Günther	gsf	1361	361c	G.Fischer
Flemming	Jutta	jup	1551	357c	J.Flemming
Fritsche	Ullrich	ufr	1821	361a	U.Fritsche
Fritzsche	Barbara	bfr	1716	357c	B.Fritzsche
Grunewald	Dietmar	dgr	1724	363c	D.Grunewald
Härtel	Gudrun	gha	1656	072	G.Haertel
Heide	Gerd	ghe	1525	363a	G.Heide
Heik	Andreas	anhe	1723	364c	A.Heik
Heine	Detlef	dhe	1553	069	D.Heine
Helbig	Rupert	rhe	1556	073	R.Helbig
Prof.Dr.Hübner	Uwe	hue	1464	357d	U.Huebner
Junge	Claudia	cho	1656	072	C.Junge
Junghänel	Jens	jju	1423	361b	J.Junghaenel
Kempe	Lothar	lke	1552	357a	L.Kempe
Kempe	Jacqueline	jke	1551	357c	J.Kempe
Lang	Thomas	tla	1823	363a	T.Lang
Meißner	Erich	erm	11.509	068b	E.Meissner
Mittelbach	Andreas	ami	1428	361a	A.Mittelbach
Muschner	Ingrid	imu	1656	072	I.Muschner
Müller	Bert	bmu	1656	072	B.Mueller
Müller	Thomas	thm	1755	362	T.Mueller
Pudlat	Rosita	rpu	1548	363c	R.Pudlat
Richter	Frank	fri	1361	361c	F.Richter
Riedel	Ursula	uri	1425	363d	U.Riedel
Dr.Riedel	Wolfgang	wri	1422	364b	W.Riedel
Schier	Thomas	schier	1423	361b	T.Schier
Schönherr	Ines	ins	1656	072	I.Schoenherr
Wegener	Edwin	ewe	11.552	364a	E.Wegener
Winkler	Jürgen	jwin	1725	364a	J.Winkler
Dr.Wolf	Ludwig	lwo	1672	361b	L.Wolf
Ziegler	Christoph	czi	1548	363c	C.Ziegler
Zscheile	Kai	kzs	1553	069	K.Zscheile

Tabelle A.1: Mitarbeiterverzeichnis des URZ

Anhang B

Beispieldokumente

B.1 Der Brief



TECHNISCHE UNIVERSITÄT CHEMNITZ
Universitätsrechenzentrum

TU Chemnitz, URZ, D-09107 Chemnitz

Prof. Dr. Hans Forschegut
Institut für Abfallbeseitigung
Fachhochschule Waldstadt
Postfach 3322
1100 Waldstadt

Ihr Zeichen, Ihre Nachricht vom	Unser Zeichen, unsere Nachricht vom	Telefon	Chemnitz, den
April, April	mamu	(0371) 531-1234	11.11.99

Sehr geehrter Herr Prof. Forschegut,

von Kollegen habe ich erfahren, daß sich bei Ihnen eine große Anzahl von Alka-Seltzer Flaschen mit nur noch einer Tablette angesammelt hat, da eine Flasche 25 Tabletten enthält, der Beipackzettel aber angibt, daß stets 2 Tabletten gleichzeitig einzunehmen sind.

Ich forsche gerade im Bereich möglicher Anwendungen einzelner Schmerztabletten. Falls Sie so freundlich wären, Ihre Alka-Seltzer Sammlung für unser Projekt zu stiften, würde ich Ihnen gerne Vorabdrucke aller zukünftigen Forschungsberichte zur Verfügung stellen, die wir über dieses kritische Problem veröffentlichen.

Mit freundlichen Grüßen

Max Müller
Leiter der Kaffestube

Dienstsitz Sekretariat URZ:
Straße der Nationen 62
Raum 357c
Telefon: (0371) 531-1551
Fax: (0371) 531-1629
Email: urz@tu-chemnitz.de

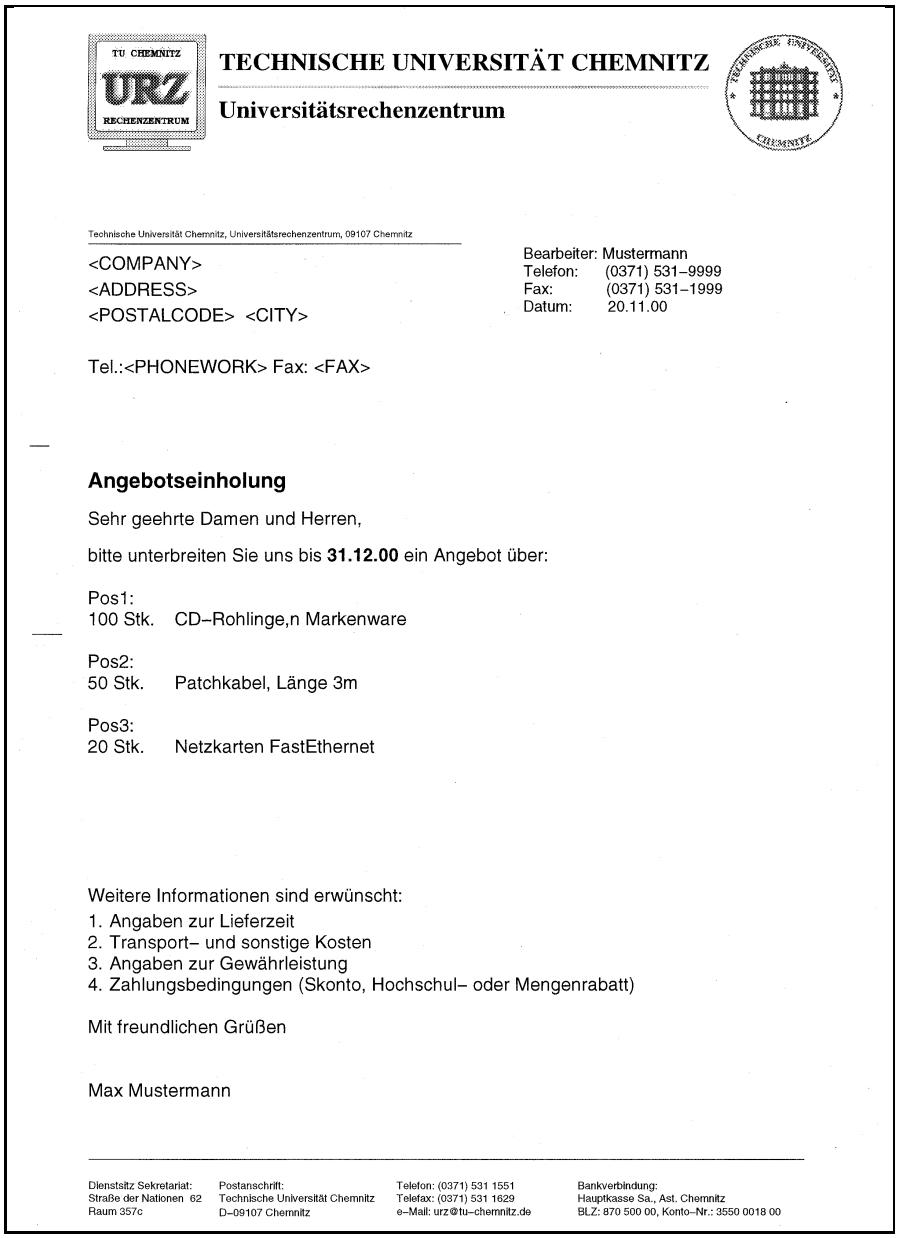
Postanschrift:
TU Chemnitz
Universitätsrechenzentrum
D-09107 Chemnitz

Paketanschrift:
TU Chemnitz
Universitätsrechenzentrum
Straße der Nationen 62
D-09111 Chemnitz

Bankverbindung der Universität:
Empfänger: Hauptkasse Sa., Ast. Chemnitz
Stadtsparkasse Dresden
BLZ: 850 551 42
Konto-Nr.: 341 301 137

Abbildung B.1: Beispiel eines Briefes

B.2 Die Angebotseinholung



TU CHEMNITZ URZ RECHENZENTRUM

TECHNISCHE UNIVERSITÄT CHEMNITZ

Universitätsrechenzentrum

Technische Universität Chemnitz, Universitätsrechenzentrum, 09107 Chemnitz

<COMPANY>
<ADDRESS>
<POSTALCODE> <CITY>

Tel.:<PHONEWORK> Fax: <FAX>

Bearbeiter: Mustermann
Telefon: (0371) 531–9999
Fax: (0371) 531–1999
Datum: 20.11.00

Angebotseinholung

Sehr geehrte Damen und Herren,

bitte unterbreiten Sie uns bis **31.12.00** ein Angebot über:

Pos1:
100 Stk. CD–Rohlinge,n Markenware

Pos2:
50 Stk. Patchkabel, Länge 3m

Pos3:
20 Stk. Netzkarten FastEthernet

Weitere Informationen sind erwünscht:
1. Angaben zur Lieferzeit
2. Transport– und sonstige Kosten
3. Angaben zur Gewährleistung
4. Zahlungsbedingungen (Skonto, Hochschul– oder Mengenrabatt)

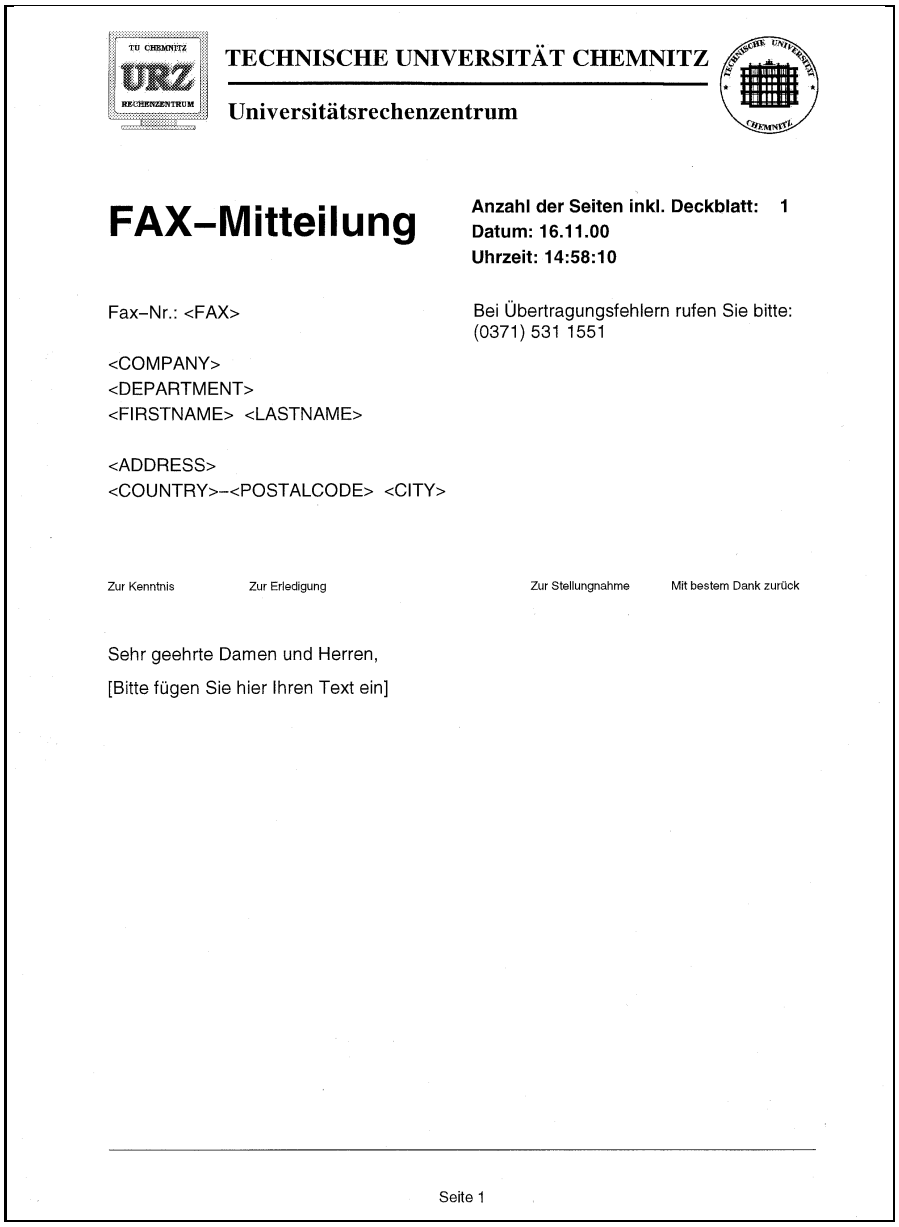
Mit freundlichen Grüßen

Max Mustermann

Dienstsitz Sekretariat: Straße der Nationen 62 Raum 357c
Postanschrift: Technische Universität Chemnitz D–09107 Chemnitz
Telefon: (0371) 531 1551 Telefax: (0371) 531 1629 e–Mail: urz@tu–chemnitz.de
Bankverbindung: Hauptkasse Sa., Ast. Chemnitz BLZ: 870 500 00, Konto–Nr.: 3550 0018 00

Abbildung B.2: Beispiel einer Angebotseinholung

B.3 Das Fax



TU CHEMNITZ URZ RECHENZENTRUM

TECHNISCHE UNIVERSITÄT CHEMNITZ

Universitätsrechenzentrum

FAX–Mitteilung

Anzahl der Seiten inkl. Deckblatt: 1
Datum: 16.11.00
Uhrzeit: 14:58:10

Fax–Nr.: <FAX>

Bei Übertragungsfehlern rufen Sie bitte:
(0371) 531 1551

<COMPANY>
<DEPARTMENT>
<FIRSTNAME> <LASTNAME>

<ADDRESS>
<COUNTRY>–<POSTALCODE> <CITY>

Zur Kenntnis Zur Erledigung Zur Stellungnahme Mit bestem Dank zurück

Sehr geehrte Damen und Herren,

[Bitte fügen Sie hier Ihren Text ein]

Seite 1

Abbildung B.3: Beispiel eines Faxes

B.4 Die Hausmitteilung

TU CHEMNITZ
URZ
RECHENZENTRUM

TU CHEMNITZ, URZ

Hausmitteilung

TECHNISCHE UNIVERSITÄT CHEMNITZ

Von: 024000, Universitätsrechenzentrum

An: Prof. Forschegut

Ihr Ansprechpartner:
M.Mustermann
☎ 1111
m.muster@hrz

Unsere Zeichen: mmu/sek

1.4.96

Sehr geehrter Herr Prof. Forschegut,

von Kollegen habe ich erfahren, daß sich bei Ihnen eine große Anzahl von Alka-Seltzer Flaschen mit nur noch einer Tablette angesammelt hat, da eine Flasche 25 Tabletten enthält, der Beipackzettel aber angibt, daß stets 2 Tabletten gleichzeitig einzunehmen sind.

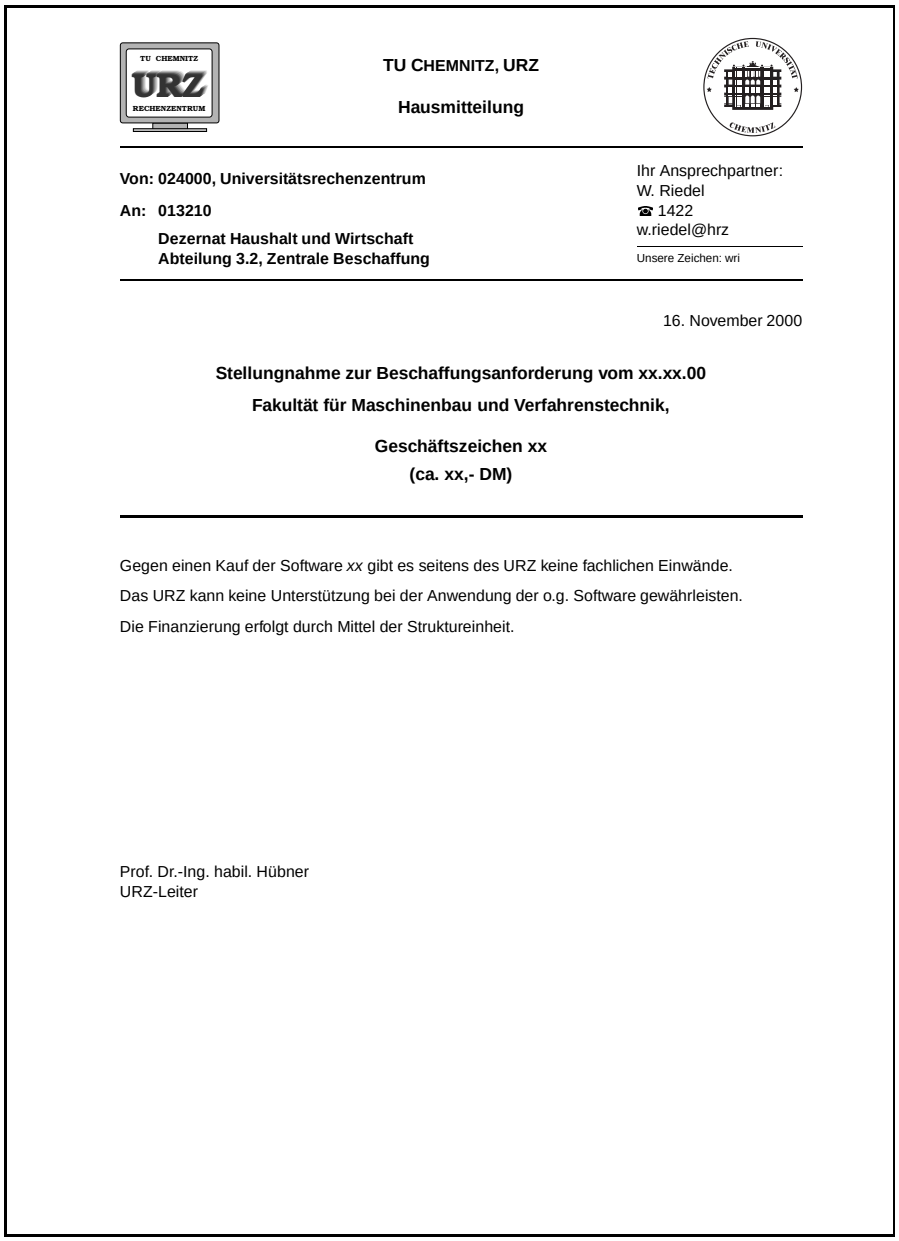
Ich forsche gerade im Bereich möglicher Anwendungen einzelner Schmerztabletten. Falls Sie so freundlich wären, Ihre Alka-Seltzer Sammlung für unser Projekt zu stiften, würde ich Ihnen gerne Vorabdrucke aller zukünftigen Forschungsberichte zur Verfügung stellen, die wir über dieses kritische Problem veröffentlichen.

Mit freundlichen Grüßen

Max Mustermann
Geschaftsführer

Abbildung B.4: Beispiel einer Hausmitteilung

B.5 Die Stellungnahme zur Beschaffungsanforderung



TU CHEMNITZ
URZ
RECHENZENTRUM

TU CHEMNITZ, URZ

Hausmitteilung

TECHNISCHE UNIVERSITÄT CHEMNITZ

Von: 024000, Universitätsrechenzentrum

An: 013210
Dezernat Haushalt und Wirtschaft
Abteilung 3.2, Zentrale Beschaffung

Ihr Ansprechpartner:
W. Riedel
☎ 1422
w.riedel@hrz

Unsere Zeichen: wri

16. November 2000

Stellungnahme zur Beschaffungsanforderung vom xx.xx.00
Fakultät für Maschinenbau und Verfahrenstechnik,

Geschäftszeichen xx
(ca. xx,- DM)

Gegen einen Kauf der Software *xx* gibt es seitens des URZ keine fachlichen Einwände.

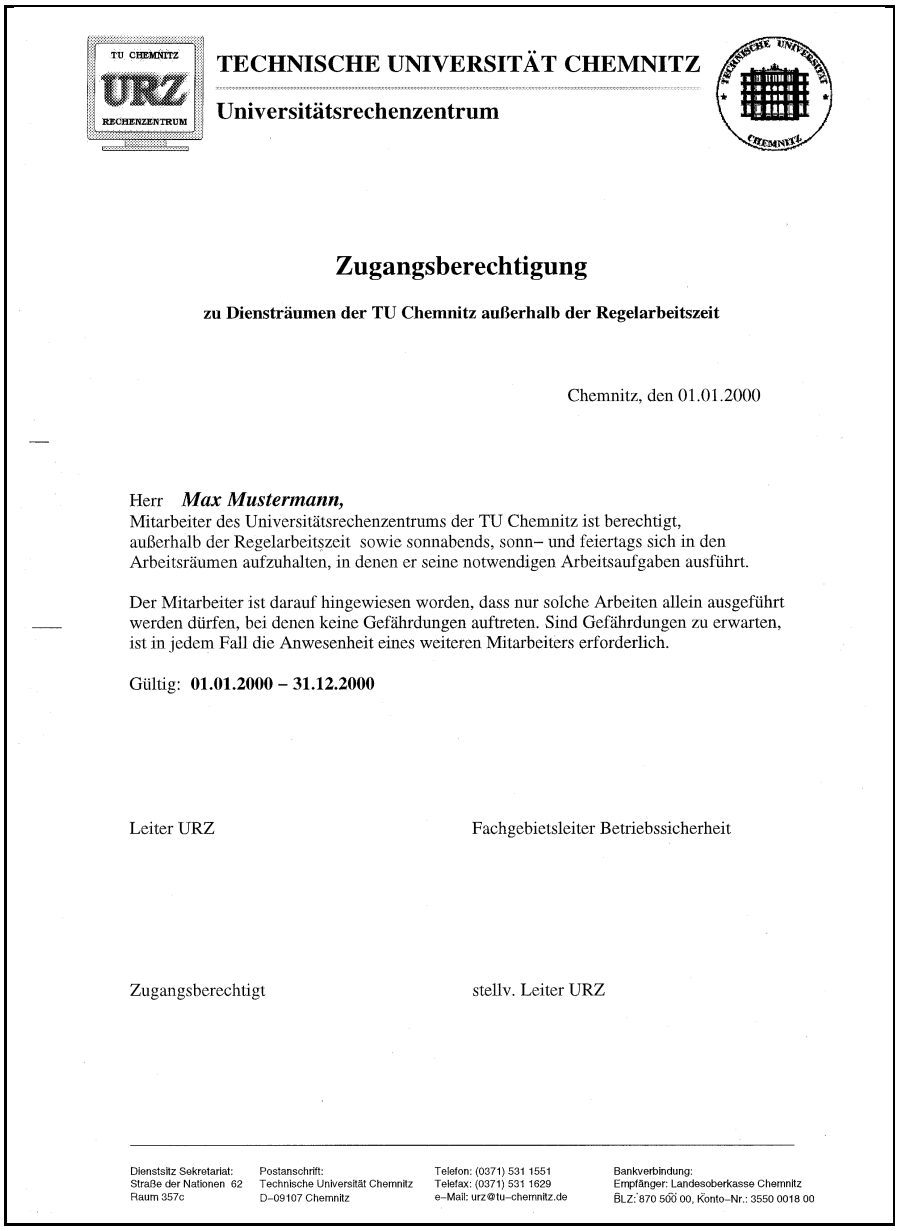
Das URZ kann keine Unterstützung bei der Anwendung der o.g. Software gewährleisten.

Die Finanzierung erfolgt durch Mittel der Struktureinheit.

Prof. Dr.-Ing. habil. Hübner
URZ-Leiter

Abbildung B.5: Beispiel eines Gutachtens

B.6 Die Zugangsberechtigung



TU CHEMNITZ URZ RECHENZENTRUM

TECHNISCHE UNIVERSITÄT CHEMNITZ

Universitätsrechenzentrum

TECHNISCHE UNIVERSITÄT CHEMNITZ

Zugangsberechtigung

zu Diensträumen der TU Chemnitz außerhalb der Regelarbeitszeit

Chemnitz, den 01.01.2000

Herr ***Max Mustermann,***
Mitarbeiter des Universitätsrechenzentrums der TU Chemnitz ist berechtigt, außerhalb der Regelarbeitszeit sowie sonnabends, sonn– und feiertags sich in den Arbeitsräumen aufzuhalten, in denen er seine notwendigen Arbeitsaufgaben ausführt.

Der Mitarbeiter ist darauf hingewiesen worden, dass nur solche Arbeiten allein ausgeführt werden dürfen, bei denen keine Gefährdungen auftreten. Sind Gefährdungen zu erwarten, ist in jedem Fall die Anwesenheit eines weiteren Mitarbeiters erforderlich.

Gültig: **01.01.2000 – 31.12.2000**

Leiter URZ

Fachgebietsleiter Betriebssicherheit

Zugangsberechtigt

stellv. Leiter URZ

Dienstsitz Sekretariat: Straße der Nationen 62 Raum 357c
Postanschrift: Technische Universität Chemnitz D–09107 Chemnitz
Telefon: (0371) 531 1551 Telefax: (0371) 531 1629 e–Mail: urz@tu–chemnitz.de
Bankverbindung: Empfänger: Landesoberkasse Chemnitz BLZ: 870 500 00, Konto–Nr.: 3550 0018 00

Abbildung B.6: Beispiel einer Zugangsberechtigung

B.7 Der Beschaffungsantrag

Bedarfsstelle
Universitätsrechenzentrum

Ort, Datum
Chemnitz, den 20.11.00

Anschrift
Universitätsrechenzentrum

Beschaffungsantrag an
den Kanzler der

TU Chemitz
–Zentrale Beschaffung–

Zuständige Bearbeiter
Herr Mustermann

Fernsprecher
0371/ 531–1716

Geschäftszeichen
024000/00999/99999–09

Nachfolgend aufgeführte Lieferungen/Leistungen werden beantragt:
(weitere Angaben – soweit erforderlich – auf gesondertem Blatt)

lfd. Nr.	Genaue Bezeichnung des Gegenstandes bzw. der Leistung (ggf. Bestell–Nr. des vorgeschlagenen Lieferanten).	benötigte Menge	veranschlagter Preis in DM je Einheit	Gesamt
01	Kugelschreiber	50	0,5	25,00 DM
02	Ordner A4	10	2,39	23,90 DM
03	Schreibblock, kar.	15	0,89	13,35 DM
04	Aktendulli	50	0,2	10,00 DM
05	Schere	5	3,99	19,95 DM
06	Klebeband	5	2,49	12,45 DM
07	Heftklammern	10	1,97	19,70 DM
		16,00%	Mwst.	19,90 DM
			Summe	**144,25 DM**

Empfänger und Bestimmungsort:
Name: Herr Mustermann, UT Str. d. Nationen 62, Zi.: 357

vorgeschlagene Lieferfirma:
Firma: <COMPANY>
<ADDRESS> <COUNTRY><POSTALCODE> <CITY>
Tel.: <PHONEWORK> Fax: <FAX>

Einbauteile, Zusatzgeräte nein

Inv–Nr. Bezeichnung des Hauptgerätes

ja, zu _____|_____

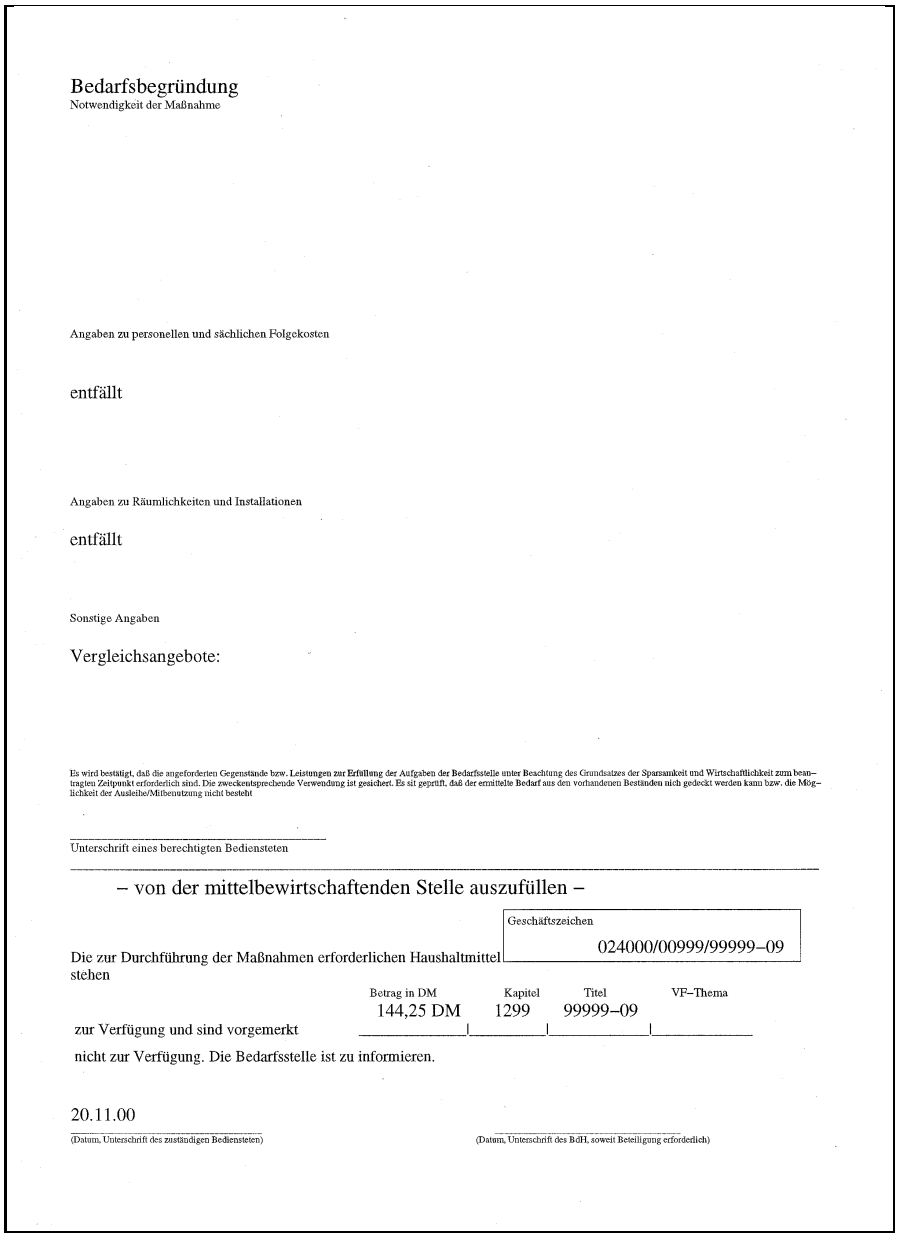
Inv–Nr. Bezeichnung des Hauptgerätes

_____|_____

Liefertermin/Lieferfrist
baldmöglichst

Ersatzteile, keine Inventarisierung!

Abbildung B.7: Beispiel eines Beschaffungsantrages Seite 1



Bedarfsbegründung
Notwendigkeit der Maßnahme

Angaben zu personellen und sächlichen Folgekosten

entfällt

Angaben zu Räumlichkeiten und Installationen

entfällt

Sonstige Angaben

Vergleichsangebote:

Es wird bestätigt, daß die angeforderten Gegenstände bzw. Leistungen zur Erfüllung der Aufgaben der Bedarfsstelle unter Beachtung des Grundsatzes der Sparsamkeit und Wirtschaftlichkeit zum beantragten Zeitpunkt erforderlich sind. Die zweckentsprechende Verwendung ist gesichert. Es sit geprüft, daß der ermittelte Bedarf aus den vorhandenen Beständen nich gedeckt werden kann bzw. die Möglichkeit der Ausleihe/Mitbenutzung nicht besteht

Unterschrift eines berechtigten Bediensteten

– von der mittelbewirtschaftenden Stelle auszufüllen –

Geschäftszeichen 024000/00999/99999–09

Die zur Durchführung der Maßnahmen erforderlichen Haushaltmittel stehen

	Betrag in DM	Kapitel	Titel	VF–Thema
zur Verfügung und sind vorgemerkt	144,25 DM	1299	99999–09	

nicht zur Verfügung. Die Bedarfsstelle ist zu informieren.

20.11.00

(Datum, Unterschrift des zuständigen Bediensteten)

(Datum, Unterschrift des BdH, soweit Beteiligung erforderlich)

Abbildung B.8: Beispiel eines Beschaffungsantrages Seite 2

Abkürzungsverzeichnis

URZ	Universitätsrechenzentrum
W3C	World Wide Web Consortium
XML	Extensible Markup Language
SGML	Standard Generalized Markup Language
DTD	Document Type Definition
DOM	Document Object Model
SAX	Simple API for XML
URL	Uniform Resource Locator
URI	Univeral Resource Identifiers
HTML	HyperText Markup Language
XHTML	Extensible HyperText Markup Language
CGI	Common Gateway Interface
DCD	Document Content Description
SOX	Schema for Object-Oriented XML
DDML	Document Definition Markup Language
PDA	Personal Digital Assisstant
DSSSL	Document Style Semantics and Specification Language
CSS	Cascading Style Sheet
XSL	Extensible Style Language
XSLT	XSL Transformations
XSL-FO	XSL Formatting Objects
RDF	Resource Description Framework
CBL	Common Business Library
cXML	Commerce XML
XFA	XML Forms Architecture
XFDL	Extensible Forms Description Language
Tcl	Tool Command Language

Literaturverzeichnis

[Rai99] Paul Raines, Jeff Tranter: *Tcl/Tk in a Nutshell*; O'Reilly & Associates, Inc., März 1999

[Bal99] Steve Ball: *Web Tcl Complete*; McGraw-Hill, 1999

[Wal99] Norman Walsh & Leonard Mueller: *DocBook - The Definitive Guide*; O'Reilly & Associates, Inc., Oktober 1999

[Mic99] Thomas Michel: *XML kompakt*; Eine praktische Einführung, Carl Hanser Verlag München Wien, 1999

[Gol99] C.F. Goldfarb, P.Prescod: *XML Handbuch*; Prentice Hall, München, 1999.

[Hue98] Uwe Hübner: *Entwurf Verteilter Systeme*; Vorlesungsmaterial, TU Chemnitz, Fakultät für Informatik, 1998

[Rie95] Wolfgang Riedel: *Einführung in LaTeX2e*; TU Chemnitz, Version 2, 12/1995

[Jet00] JetForm (Deutschland) GmbH: *Elektronische Formulare zur Automatisierung von Büro- und Verwaltungsprozessen*; JetForm Referenzkunden und Fallstudien, 2000

[Sch99] Klaus Schlüter: *Websperanto*; Gateway 02/99 S.56ff, Computerwoche Verlag GmbH

[DIN676] Deutsches Institut für Normung e.V.: *Geschäftsbrief*; DIN 676, Mai 1995

[DIN5008] Deutsches Institut für Normung e.V.: *Schreib- und Gestaltungsregeln für die Textverarbeitung*; DIN 5008, Mai 1996

[Fra99] S.Franke, K.Plessner: *EU-Richtlinie für digitale Signaturen*; NetworkWorld 5/6-99 S.4, Computerwoche Verlag GmbH

[Gar00] GartnerGroup RAS Service: *Printing Myths - E-Business Eliminates Paper*; Reserch Note, 25. Januar 2000

[Lib96] Don Libes: *Writing CGI Scripts in Tcl*; 1996, http://www.cme.nist.gov/msid/pubs/libes96c.ps

[And98] Jörg Anders, (Werkzeug zur Generierung von Kanzlerbeschaffungsanforderungen, Version 2), TU Chemnitz, http://rnvs.informatik.tu-chemnitz.de/kanzanf/kanzanf.html

[Hic98] Tony Hicks: *Shoud We Be Using ISO 12083?*; The Journal of Electronic Publishing, Juni 1998, Volume 3, Issue 4, http://www.press.umich.edu/jep/03-04/hicks.html

[Gei98] Ivo Geis: *Rechtsfragen des elektronischen Geschäftsverkehrs*; Referat, 03. September 1998, http://www.ivo-geis.de/documents/elektrgeschaeftsverkehr.html

[Bra98] Tim Bray: *Annotated XML Specification*; 1998, http://www.xml.com/axml/axml.html

[XML] T.Bray, J.Paoli, C.M.Sperberg-McQueen, E.Maler: *Extensible Markup Language (XML) 1.0 (Second Edition)*; W3C Recommendation 6 October 2000, http://www.w3.org/TR/2000/REC-xml-20001006

[SGML] Standard Generalized Markup Language (SGML), ISO-Norm 8879:1986.
Robin Cover: *SGML: General Introductions and Overviews*; http://www.oasis-open.org/cover/general.html

[DOM] World Wide Web Consortium: *Document Object Model (DOM) Level 1 Specification*; Version 1.0, W3C Recommendation 1 October, 1998, http://www.w3.org/TR/REC-DOM-Level-1/

[XML-Schema] World Wide Web Consortium: *XML Schema Part 1: Structures*; W3C Working Draft 7 April 2000, http://www.w3.org/TR/2000/WD-xmlschema-1-20000407/
World Wide Web Consortium: *XML Schema Part 2: Datatypes*; W3C Working Draft 07 April 2000, http://www.w3.org/TR/2000/WD-xmlschema-2-20000407/

[SAX] David Megginson: *The Simple API for XML*; http://www.megginson.com/SAX/index.html

[DSSSL] ISO/IEC: *Document Style Semantics and Specification Language (DSSSL)*; ISO/IEC 10179:1996, 1996, ftp://ftp.ornl.gov/pub/sgml/WG8/DSSSL/dsssl96b.pdf

[CSS] H.W.Lie, B.Bos: *Cascading Style Sheets, level 1*; W3C Recommendation 17 Dec 1996, http://www.w3.org/TR/REC-CSS1.html
H.W.Lie, B.Bos, C.Lilley, I.Jacobs: *Cascading Style Sheets, level 2* ; W3C Recommendation 12-May-1998, http://www.w3.org/TR/REC-CSS2/

[XSL] World Wide Web Consortium: *Extensible Stylesheet Language (XSL) Version 1.0*; W3C Working Draft 18 October 2000, http://www.w3.org/TR/2000/WD-xsl-20001018/

[XSLT] World Wide Web Consortium: *XSL Transformations (XSLT) Version 1.0*; W3C Recommendation 16 November 1999, http://www.w3.org/TR/xslt.html

[XPath] World Wide Web Consortium: *XML Path Language (XPath) Version 1.0*; W3C Recommendation, 16 November 1999, http://www.w3.org/TR/1999/REC-xpath-19991116

[Jade] James Clark: *Jade - James' DSSSL Engine*; http://www.jclark.com/jade/

[DBStyle] Norman Walsh: *The Modular DocBook Stylesheets*; http://nwalsh.com/dicbook/dsssl/

[XHTML] World Wide Web Consortium: *XHTML 1.0: The Extensible HyperText Markup Language*; A Reformulation of HTML 4 in XML 1.0, W3C Recommendation 26 Januar 2000, http://www.w3.org/TR/2000/REC-xhtml1-20000126

[HTML] World Wide Web Consortium: *HTML 4.01 Specification*; W3C Recommendation 24 December 1999, http://www.w3.org/TR/html4/

[RDF] O.Lassila, R.R.Swick: *Resource Description Framework (RDF) Model and Syntax Specification*; W3C Recommendation 22 February 1999, http://www.w3.org/TR/REC-rdf-syntax/

[RDFSchema] D.Brickley, R.V.Guha: *Resource Description Framework (RDF) Schema Specification 1.0*; W3C Candidate Recommendation 27 March 2000, http://www.w3.org/TR/2000/CR-rdf-schema-20000327/

[OEB] Open eBook Initiative: *Open eBook: Publication Structure 1.0*; 16. September 1999, http://www.openebook.org/OEB1.html

[TEI] Text Encoding Initiative: *Guidelines for Electronic Text Encoding and Interchange*; http://www.hti.umich.edu/docs/TEI/

[CBL] Commerce One, Inc.: *Structure Reference: Common Business Library v. 2.0, rel. 3*; 1999, http://www.commerceone.com/xml/cbl/docs/reference.html

[cXML] Ariba, Inc.: *cXML/1.0*; 1999, http://www.cXML.org/files/cxml.pdf

[BizTalk] Microsoft Corp.: *BizTalk Framework 1.0*; 1999, http://www.biztalk.org/Resources/BTF1_0.doc

[tpaML] IBM, Trading Partner Agreement Markup Language, http://www-4.ibm.com/software/developer/library/tpaml.html

[XForms] World Wide Web Consortium: *XForms 1.0: Data Model*; W3C Working Draft, 06. April 2000, http://www.w3.org/TR/2000/WD-xforms-datamodel-20000406

[XFA] JetForm Corporation: *XML Forms Architecture (XFA)*; XFA-Template, http://www.xfa.com/spec/xfa-template/xfa-template.html

[XFDL] J.Boyer, T.Bray, M.Gordon: *Extensible Forms Description Language (XFDL) 4.0*; W3C Note, September 2, 1998, http://www.w3.org/TR/NOTE-XFDL

[SigG] Die Bundesregierung: *Gesetz zur digitalen Signatur*; (Signaturgesetz - SigG), 13. Juni 1997, http://www.regtp.de/imperia/md/content/tech_reg_t/digisign/4.pdf

[EUSig] Das Europäische Parlament und der Rat der europäischen Union: *Richtlinie 1999/93/EG*; Gemeinschaftliche Rahmenbedingungen für elekronische Signaturen, 13. Dezember 1999, http://www.bmwi.de/download/eu-signatur-d.pdf

[DSig] World Wide Web Consortium: *XML-Signature Syntax and Processing*; W3C Working Draft, 01. Juni 2000, http://www.w3.org/TR/2000/WD-xmldsig-core-20000601

[CanXML] World Wide Web Consortium: *Canonical XML Version 1.0*; W3C Working Draft, 13. Juni 2000, http://www.w3.org/TR/2000/WD-xml-c14n-20000613

[CGI] The Common Gateway Interface, http://hoohoo.ncsa.uiuc.edu/cgi/

[Kus99] J.J.Kuslich: *Introduction to JavaServer Pages*; Server-Side Scripting the Java Way, 6/99, http://developer.iplanet.com:80/viewsource/kuslich_jsp/kuslich_jsp_p.html

[ModTcl] Apache Tcl Project: http://tcl.apache.org/mod_dtcl/

[Kri97] D. Kristol, L. Montulli: *HTTP State Management Mechanism*; RFC2109, Februar 1997, http://www.cis.ohio-state.edu/htbin/rfc/rfc2109.html

[TclDOM] Zveno Pty Ltd: *TclDOM Reference*; http://www.zveno.com/zm.cgi/in-tclxml/in-tcldom/TclDOM-1.6/docs/reference.tml

[TclXML] Zveno Pty Ltd: *TclXML Parsers*; http://www.zveno.com/zm.cgi/in-tclxml/TclXML-1.2/docs/reference.tml

[TclXSL] Zveno Pty Ltd: *Tcl and XSL*; http://www.zveno.com/zm.cgi/in-tclxml/in-xsl.tml

[Har00] E.R.Harold: *Chapter 14 of the XML Bible: XSL Transformations*; 2000, http://www.ibiblio.org/xml/books/bible/updates/14.html

[GnuPG] Free Software Foundation, Inc.: *Das GNU-Handbuch zum Schutze der Privatsphäre*; 2000, http://www.gnupg.org/gph/de/manual/

Aufgabenstellung

Thema: Büroarbeit im Intranet

Ziel der Arbeit ist die Schaffung einer modernen Infrastruktur aus attraktiven und nutzungssicheren Applikationen zur Erledigung typischer Geschäftsvorfälle. Die Nutzerschnittstelle sollte Web-basiert sein, die interne Verarbeitung auf XML beruhen.
Dabei sollen die notwendigen systemtechnischen und technologischen Aspekte des realen Einsatzes im URZ berücksichtigt werden.

Arbeitsschritte

1. Analyse laufender Geschäftsvorfälle zwecks Bildung von Dokumentenklassen und der Definition deren charakteristischer Merkmale (bzgl. Struktur und Layout). Als Minimum sind dabei Briefe, Faxe, Hausmitteilungen, Angebotseinholungen und Beschaffungsgutachten zu betrachten.
2. Analyse der Fähigkeiten aktueller XML-Applikationen zur Erzeugung und Verarbeitung druckfähiger Dokumente (Open E-Book, DocBook ...)
3. Entwicklung der Menge notwendiger Tags pro Dokumentenklasse
4. Entwurf der Nutzerschnittstellen des Systems
5. Entwurf der internen Datenstruktur und des Konzepts zur Speicherung der Dokumente
6. Analyse der Verwendbarkeit der Prototyp-Web-Formulare
7. Realisierung der Werkzeuge
8. Erprobung

Betreuender Hochschullehrer: Prof. Dr. Uwe Hübner
Betreuer im URZ: Dr. Wolfgang Riedel

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbständig und ohne unzulässig Hilfe verfasst und keine anderen als die Quellen und Hilfsmittel verwendet habe.

Chemnitz, den 30. November 2000